# **PyNLPI Documentation**

Release 1.2.8

**Maarten van Gompel** 

## Contents

<ul> <li>2 Data Types</li> <li>3 Evaluation &amp; Experiments</li> <li>4 FoLiA library <ul> <li>4.1 Reading FoLiA</li> <li>4.1.1 Loading a document</li> <li>4.1.2 Printing text</li> <li>4.1.3 Index</li> <li>4.1.4 Elements</li> </ul> </li> </ul>	3
4 FoLiA library 4.1 Reading FoLiA	5
4.1       Reading FoLiA          4.1.1       Loading a document          4.1.2       Printing text          4.1.3       Index	9
4.1.1 Loading a document	13
4.1.2       Printing text	13
4.1.3 Index	13
	21
4.1.4 Elements	22
	22
4.1.5 Obtaining list of elements	94
4.1.6 Select method	95
4.1.7 Selection Shortcuts	95
4.1.8 Navigating a document	96
4.1.9 Structure Annotation Types	97
4.1.10 Common attributes	425
4.1.11 Annotations	425
4.2 Editing FoLiA	797
4.2.1 Creating a new document	797
4.2.2 Declarations	797
4.2.3 Adding structure	797
4.2.4 Adding annotations	798
4.2.5 Adding span annotation	799
4.2.6 Deleting annotations	800
4.2.7 Copying annotations	800
4.3 Searching in a FoLiA document	801
4.3.1 Corpus Query Language (CQL)	801
4.3.2 FoLiA Query Language (FQL)	802
4.3.3 Streaming Reader	804
4.4 Higher-Order Annotations	806
4.4.1 Text Markup	806
4.4.2 Features	859
4.4.3 Alternatives	915
4.4.4 Corrections	940
4.4.5 Alignments	1006

	4.5	4.4.6 Descriptions, Metrics	
5	Forn		053
	5.1	Corpus Gesproken Nederlands	
	5.2	FoLiA	
	5.3	GIZA++	
	5.4	Moses	
	5.5	SoNaR	
	5.6 5.7	Taggerdata	
	3.7	TiMBL	1033
6	Lang	guage Models	057
7	Sear	ch Algorithms	059
8	Statis	stics and Information Theory	061
	8.1	Generic functions	1061
	8.2	Frequency Lists and Distributions	1061
	8.3	API Reference	1062
9	Text	Processors	065
	9.1	Tokenisation	1065
	9.2	N-gram extraction	1065
10	Indic	ees and tables	069
Рy	thon I	Module Index	071

PyNLPl, pronounced as 'pineapple', is a Python library for Natural Language Processing. It contains various modules useful for common, and less common, NLP tasks. PyNLPl can be used for basic tasks such as the extraction of n-grams and frequency lists, and to build simple language model. There are also more complex data types and algorithms. Moreover, there are parsers for file formats common in NLP (e.g. FoLiA/Giza/Moses/ARPA/Timbl/CQL). There are also clients to interface with various NLP specific servers. PyNLPl most notably features a very extensive library for working with FoLiA XML (Format for Linguistic Annotatation).

The library is a divided into several packages and modules. It works on Python 2.7, as well as Python 3.

The following modules are available:

- pynlpl.datatypes Extra datatypes (priority queues, patterns, tries)
- pynlpl.evaluation Evaluation & experiment classes (parameter search, wrapped progressive sampling, class evaluation (precision/recall/f-score/auc), sampler, confusion matrix, multithreaded experiment pool)
- pynlpl.formats.cgn Module for parsing CGN (Corpus Gesproken Nederlands) part-of-speech tags
- pynlpl.formats.folia Extensive library for reading and manipulating the documents in FoLiA format (Format for Linguistic Annotation).
- pynlpl.formats.fql Extensive library for the FoLiA Query Language (FQL), built on top of pynlpl. formats.folia. FQL is currently documented here.
- pynlpl.formats.cql Parser for the Corpus Query Language (CQL), as also used by Corpus Workbench and Sketch Engine. Contains a convertor to FQL.
- pynlpl.formats.giza Module for reading GIZA++ word alignment data
- pynlpl.formats.moses Module for reading Moses phrase-translation tables.
- pynlpl.formats.sonar Largely obsolete module for pre-releases of the SoNaR corpus, use pynlpl. formats.folia instead.
- pynlpl.formats.timbl Module for reading Timbl output (consider using python-timbl instead though)
- pynlpl.lm.lm- Module for simple language model and reader for ARPA language model data as well (used by SRILM).
- pynlpl.search Various search algorithms (Breadth-first, depth-first, beam-search, hill climbing, A star, various variants of each)
- pynlpl.statistics Frequency lists, Levenshtein, common statistics and information theory functions
- pynlpl.textprocessors Simple tokeniser, n-gram extraction

#### Contents:

Contents 1

2 Contents

## CHAPTER 1

### **Common Functions**

## CHAPTER 2

### **Data Types**

This library contains various extra data types, based to a certain extend on MIT-licensed code from Peter Norvig, AI: A Modern Appproach: http://aima.cs.berkeley.edu/python/utils.html

```
class pynlpl.datatypes.FIFOQueue (data=[])
     A First-In-First-Out Queue
     append(item)
     extend(items)
         Append all elements from items to the queue
     pop()
         Retrieve the next element in line, this will remove it from the queue
class pynlpl.datatypes.Pattern(data, classdecoder=None)
     static fromstring(s, classencoder)
     iterbytes (begin=0, end=0)
class pynlpl.datatypes.PatternMap(default=None)
     items()
class pynlpl.datatypes.PatternSet
     add (pattern)
     remove (pattern)
class pynlpl.datatypes.PriorityQueue(data=[],f=<function PriorityQueue.<lambda>>, min-
                                              imize=False, length=0, blockworse=False, blocke-
                                              qual=False, duplicates=True)
     A queue in which the maximum (or minumum) element is returned first, as determined by either an external
```

score function f (by default calling the objects score() method). If minimize=True, the item with minimum f(x)

is returned first; otherwise is the item with maximum f(x) or x.score().

length can be set to an integer > 0. Items will only be added to the queue if they're better or equal to the worst scoring item. If set to zero, length is unbounded. blockworse can be set to true if you want to prohibit adding worse-scoring items to the queue. Only items scoring better than the *BEST* one are added. blockequal can be set to false if you also want to prohibit adding equally-scoring items to the queue. (Both parameters default to False)

```
append (item)
```

Adds an item to the priority queue (in the right place), returns True if successfull, False if the item was blocked (because of a bad score)

#### pop()

Retrieve the next element in line, this will remove it from the queue

#### prune(n)

prune all but the first (=best) n items

#### prunebyscore (score, retainequalscore=False)

Deletes all items below/above a certain score from the queue, depending on whether minimize is True or False. Note: It is recommended (more efficient) to use blockworse=True / blockequal=True instead! Preventing the addition of 'worse' items.

#### randomprune (n)

prune down to n items at random, disregarding their score

#### score(i)

Return the score for item x (cheap lookup), Item 0 is always the best item

#### stochasticprune(n)

prune down to n items, chance of an item being pruned is reverse proportional to its score

```
class pynlpl.datatypes.Queue
```

**Queue is an abstract class/interface. There are three types:** Python List: A Last In First Out Queue (no Queue object necessary). FIFOQueue(): A First In First Out Queue. PriorityQueue(lt): Queue where items are sorted by lt, (default <).

Each type supports the following methods and functions: q.append(item) – add an item to the queue q.extend(items) – equivalent to: for item in items: q.append(item) q.pop() – return the top item from the queue len(q) – number of items in q (also q.\_\_len()).

#### extend(items)

Append all elements from items to the queue

```
class pynlpl.datatypes.Tree(value=None, children=None)
```

Simple tree structure. Nodes are themselves trees.

#### append (item)

Add an item to the Tree

#### leaf()

Is this a leaf node or not?

#### class pynlpl.datatypes.Trie(sequence=None)

Simple trie structure. Nodes are themselves tries, values are stored on the edges, not the nodes.

```
append (sequence)
```

#### depth()

Returns the depth of the current node

#### find (sequence)

#### items()

```
leaf()
    Is this a leaf node or not?

path()
    Returns the path to the current node

root()
    Returns True if this is the root of the Trie

sequence()

size()
    Size is number of nodes under the trie, including the current node

walk (leavesonly=True, maxdepth=None, _depth=0)
    Depth-first search, walking through trie, returning all encounterd nodes (by default only leaves)
```

## CHAPTER 3

## **Evaluation & Experiments**

```
class pynlpl.evaluation.AbstractExperiment (inputdata=None, **parameters)
     defaultparameters()
     delete()
     done (warn=True)
          Is the subprocess done?
     duration()
     run()
     sample(size)
          Return a sample of the input data
     score()
     start()
          Start as a detached subprocess, immediately returning execution to caller.
     startcommand (command, cwd, stdout, stderr, *arguments, **parameters)
     wait()
class pynlpl.evaluation.ClassEvaluation(goals=[],
                                                                observations=[],
                                                                                   missing = \{\},
                                                  encoding='utf-8')
     accuracy (cls=None)
     append (goal, observation)
     auc (cls=None, macro=False)
     compute()
     confusionmatrix(casesensitive=True)
     fp_rate(cls=None, macro=False)
```

```
fscore (cls=None, beta=1, macro=False)
     outputmetrics()
     precision (cls=None, macro=False)
     recall (cls=None, macro=False)
     specificity (cls=None, macro=False)
     tp rate (cls=None, macro=False)
                                                                                       dovalida-
class pynlpl.evaluation.ConfusionMatrix(tokens=None,
                                                                  casesensitive=True,
                                                   tion=True)
     Confusion Matrix
class pynlpl.evaluation.ExperimentPool(size)
     append (experiment)
     poll (haltonerror=True)
     run (haltonerror=True)
     start (experiment)
class pynlpl.evaluation.OrdinalEvaluation(goals=[],
                                                                  observations=[],
                                                                                    missing=\{\},
                                                      encoding='utf-8')
     compute()
     mae (cls=None)
     rmse (cls=None)
class pynlpl.evaluation.ParamSearch (experiment class, input data, parameter scope, pool size=1,
                                              constraintfunc=None, delete=True)
     A simpler version of ParamSearch without Wrapped Progressive Sampling
exception pynlpl.evaluation.ProcessFailed
class pynlpl.evaluation.WPSParamSearch (experimentclass, inputdata, size, parameterscope,
                                                  poolsize=1, sizefunc=None, prunefunc=None, con-
                                                  straintfunc=None, delete=True)
     ParamSearch with support for Wrapped Progressive Sampling
     searchbest()
     test (i=None)
pynlpl.evaluation.auc(x, y, reorder=False)
     Compute Area Under the Curve (AUC) using the trapezoidal rule
     This is a general fuction, given points on a curve. For computing the area under the ROC-curve, see
     auc score().
          Parameters
                • \mathbf{x} (array, shape = [n]) - \mathbf{x} coordinates.
                • y(array, shape = [n]) - y coordinates.
                • reorder (boolean, optional (default=False)) - If True, assume that the
                 curve is ascending in the case of ties, as for an ROC curve. If the curve is non-ascending,
                 the result will be wrong.
```

Returns auc

#### Return type float

#### **Examples**

```
>>> import numpy as np
>>> from sklearn import metrics
>>> y = np.array([1, 1, 2, 2])
>>> pred = np.array([0.1, 0.4, 0.35, 0.8])
>>> fpr, tpr, thresholds = metrics.roc_curve(y, pred, pos_label=2)
>>> metrics.auc(fpr, tpr)
0.75
```

#### See also:

auc\_score() Computes the area under the ROC curve

```
pynlpl.evaluation.filesampler(files, testsetsize=0.1, devsetsize=0, trainsetsize=0, outputdir=", encoding='utf-8')
```

Extract a training set, test set and optimally a development set from one file, or multiple *interdependent* files (such as a parallel corpus). It is assumed each line contains one instance (such as a word or sentence for example).

```
pynlpl.evaluation.mae (absolute_error_values)
pynlpl.evaluation.rmse (squared_error_values)
```

## CHAPTER 4

FoLiA library

This tutorial will introduce the **FoLiA Python library**, part of PyNLPl. The FoLiA library provides an Application Programming Interface for the reading, creation and manipulation of FoLiA XML documents. The library works under Python 2.7 as well as Python 3, which is the recommended version. The samples in this documentation follow Python 3 conventions.

Prior to reading this document, it is recommended to first read the FoLiA documentation itself and familiarise yourself with the format and underlying paradigm. The FoLiA documentation can be found on the FoLiA website. It is especially important to understand the way FoLiA handles sets/classes, declarations, common attributes such as annotator/annotatortype and the distinction between various kinds of annotation categories such as token annotation and span annotation.

This Python library is also the foundation of the FoLiA Tools collection, which consists of various command line utilities to perform common tasks on FoLiA documents. If you're merely interested in performing a certain common task, such as a single query or conversion, you might want to check there if it contains is a tool that does what you want already.

### 4.1 Reading FoLiA

#### 4.1.1 Loading a document

Any script that uses FoLiA starts with the import:

```
from pynlpl.formats import folia
```

At the basis of any FoLiA processing lies the following class:

Document	This is the FoLiA Document and holds all its data in
	memory.

#### pynlpl.formats.folia.Document

class pynlpl.formats.folia.Document(\*args, \*\*kwargs)
 Bases: object

This is the FoLiA Document and holds all its data in memory.

All FoLiA elements have to be associated with a FoLiA document. Besides holding elements, the document may hold metadata including declarations, and an index of all IDs.

### **Method Summary**

init(*args, **kwargs)	Start/load a FoLiA document:
add(text)	Alias for Document.append()
alias(annotationtype, set[, fallback])	Return the alias for a set (if applicable, returns the
• • • • • • • • • • • • • • • • • • • •	unaltered set otherwise iff fallback is enabled)
append(text)	Add a text (or speech) to the document:
count(Class[, set, recursive, ignore])	See AbstractElement.count()
create(Class, *args, **kwargs)	Create an element associated with this Document.
date([value])	Get or set the document's date from/in the metadata.
declare(annotationtype, set, **kwargs)	Declare a new annotation type to be used in the doc-
	ument.
declared(annotationtype, set)	Checks if the annotation type is present (i.e.
defaultannotator(annotationtype[, set])	Obtain the default annotator for the specified annota-
•	tion type and set.
defaultannotatortype(annotationtype[, set])	Obtain the default annotator type for the specified an-
	notation type and set.
defaultdatetime(annotationtype[, set])	Obtain the default datetime for the specified annota-
· ·	tion type and set.
defaultset(annotationtype)	Obtain the default set for the specified annotation
	type.
findwords(*args, **kwargs)	
items()	Returns a depth-first flat list of all items in the docu-
	ment
json()	Serialise the document to a dict ready for seriali-
	sation to JSON.
jsondeclarations()	Return all declarations in a form ready to be seri-
	alised to JSON.
language([value])	No arguments: Get the document's language (ISO-
	639-3) from metadata Argument: Set the document's
	language (ISO-639-3) in metadata
license([value])	No arguments: Get the document's license from
	metadata Argument: Set the document's license in
	metadata
load(filename)	Load a FoLiA XML file.
paragraphs([index])	Return a generator of all paragraphs found in the
	document.
parsemetadata(node)	Internal method to parse metadata
parsesubmetadata(node)	
<pre>parsexml(node[, ParentClass])</pre>	Internal method.
parsexmldeclarations(node)	Internal method to parse XML declarations
	Continued on next page

Table 2 – continued from previous page

pendingvalidation([warnonly])	Perform any pending validations
publisher([value])	No arguments: Get the document's publisher from
pastioner ([··········])	metadata Argument: Set the document's publisher in
	metadata
save([filename])	Save the document to file.
select(Class[, set, recursive, ignore])	See AbstractElement.select()
sentences([index])	Return a generator of all sentence found in the docu-
	ment.
setimdi(node)	OBSOLETE
text([cls, retaintokenisation])	Returns the text of the entire document (returns a uni-
	code instance)
title([value])	Get or set the document's title from/in the metadata
unalias(annotationtype, alias)	Return the set for an alias (if applicable, raises an
	exception otherwise)
words([index])	Return a generator of all active words found in the
	document.
xml()	Serialise the document to XML.
xmldeclarations()	Internal method to generate XML nodes for all dec-
	larations
xmlmetadata()	Internal method to serialize metadata to XML
xmlstring()	Return the XML representation of the document as a
	string.
xpath(query)	Run Xpath expression and parse the resulting ele-
	ments.

#### **Attributes**

IDSEPARATOR

#### **Method Details**

\_\_init\_\_\_(\*args, \*\*kwargs)

Start/load a FoLiA document:

There are four sources of input for loading a FoLiA document:

1. Create a new document by specifying an *ID*:

```
doc = folia.Document(id='test')
```

2. Load a document from FoLiA or D-Coi XML file:

```
doc = folia.Document(file='/path/to/doc.xml')
```

3. Load a document from an XML string:

```
doc = folia.Document(string='<FoLiA>....</FoLiA>')
```

4. Load a document by passing a parse xml tree (lxml.etree):

doc = folia.Document(tree=xmltree)

Additionally, there are three modes that can be set with the mode= keyword argument:

4.1. Reading FoLiA

- folia.Mode.MEMORY The entire FoLiA Document will be loaded into memory. This is the default mode and the only mode in which documents can be manipulated and saved again.
- folia.Mode.XPATH The full XML tree will still be loaded into memory, but conversion to FoLiA classes occurs only when queried. This mode can be used when the full power of XPath is required.

#### **Keyword Arguments**

- **setdefinition** (dict) A dictionary of set definitions, the key corresponds to the set name, the value is a SetDefinition instance
- loadsetdefinitions (bool) download and load set definitions (default: False)
- **deepvalidation** (bool) Do deep validation of the document (default: False), implies loadsetdefinitions
- **textvalidation** (bool) Do validation of text consistency (default: False)"
- **preparsexmlcallback** (function) Callback for a function taking one argument (node, an lxml node). Will be called whenever an XML element is parsed into FoLiA. The function should return an instance inherited from folia. AbstractElement, or None to abort parsing this element (and all its children)
- parsexmlcallback (function) Callback for a function taking one argument (element, a FoLiA element). Will be called whenever an XML element is parsed into FoLiA. The function should return an instance inherited from folia. Abstract Element, or None to abort adding this element (and all its children)
- **debug** (bool) Boolean to enable/disable debug

```
__init___(*args, **kwargs)
```

Start/load a FoLiA document:

There are four sources of input for loading a FoLiA document:

1. Create a new document by specifying an *ID*:

```
doc = folia.Document(id='test')
```

2. Load a document from FoLiA or D-Coi XML file:

```
doc = folia.Document(file='/path/to/doc.xml')
```

3. Load a document from an XML string:

```
doc = folia.Document(string='<FoLiA>....</FoLiA>')
```

4. Load a document by passing a parse xml tree (lxml.etree):

```
doc = folia.Document(tree=xmltree)
```

Additionally, there are three modes that can be set with the mode= keyword argument:

- folia.Mode.MEMORY The entire FoLiA Document will be loaded into memory. This is the default mode and the only mode in which documents can be manipulated and saved again.
- folia.Mode.XPATH The full XML tree will still be loaded into memory, but conversion to FoLiA classes occurs only when queried. This mode can be used when the full power of XPath is required.

#### **Keyword Arguments**

- **setdefinition** (dict) A dictionary of set definitions, the key corresponds to the set name, the value is a SetDefinition instance
- loadsetdefinitions (bool) download and load set definitions (default: False)
- **deepvalidation** (bool) Do deep validation of the document (default: False), implies loadsetdefinitions
- **textvalidation** (bool) Do validation of text consistency (default: False)"
- **preparsexmlcallback** (function) Callback for a function taking one argument (node, an lxml node). Will be called whenever an XML element is parsed into FoLiA. The function should return an instance inherited from folia. Abstract Element, or None to abort parsing this element (and all its children)
- parsexmlcallback (function) Callback for a function taking one argument (element, a FoLiA element). Will be called whenever an XML element is parsed into FoLiA. The function should return an instance inherited from folia. AbstractElement, or None to abort adding this element (and all its children)
- **debug** (bool) Boolean to enable/disable debug

#### add (text)

Alias for Document . append ()

#### alias (annotationtype, set, fallback=False)

Return the alias for a set (if applicable, returns the unaltered set otherwise iff fallback is enabled)

#### append (text)

Add a text (or speech) to the document:

#### Example 1:

```
doc.append(folia.Text)
```

**Example 2::** doc.append( folia.Text(doc, id='example.text') )

#### Example 3:

```
doc.append(folia.Speech)
```

#### count (Class, set=None, recursive=True, ignore=True)

See AbstractElement.count()

```
create (Class, *args, **kwargs)
```

Create an element associated with this Document. This method may be obsolete and removed later.

#### date (value=None)

Get or set the document's date from/in the metadata.

No arguments: Get the document's date from metadata Argument: Set the document's date in metadata

#### declare (annotationtype, set, \*\*kwargs)

Declare a new annotation type to be used in the document.

Keyword arguments can be used to set defaults for any annotation of this type and set.

#### **Parameters**

• annotationtype — The type of annotation, this is conveyed by passing the corresponding annotation class (such as *PosAnnotation* for example), or a member of AnnotationType, such as AnnotationType.POS.

• **set** (str) – the set, should formally be a URL pointing to the set definition

#### **Keyword Arguments**

- annotator (str) Sets a default annotator
- annotatortype Should be either AnnotatorType.MANUAL or AnnotatorType.AUTO, indicating whether the annotation was performed manually or by an automated process.
- datetime (datetime.datetime) Sets the default datetime
- alias (str) Defines alias that may be used in set attribute of elements instead of the full set name

#### Example:

#### declared (annotationtype, set)

Checks if the annotation type is present (i.e. declared) in the document.

#### **Parameters**

- annotationtype The type of annotation, this is conveyed by passing the corresponding annotation class (such as *PosAnnotation* for example), or a member of AnnotationType, such as AnnotationType.POS.
- **set** (str) the set, should formally be a URL pointing to the set definition (aliases are also supported)

#### Example:

```
if doc.declared(folia.PosAnnotation, 'http://some/path/brown-tag-set'):
    ...
```

#### Returns bool

#### defaultannotator (annotationtype, set=None)

Obtain the default annotator for the specified annotation type and set.

#### **Parameters**

- annotationtype The type of annotation, this is conveyed by passing the corresponding annotation class (such as *PosAnnotation* for example), or a member of AnnotationType, such as AnnotationType.POS.
- **set** (str) the set, should formally be a URL pointing to the set definition

#### **Returns** the set (str)

**Raises** NoDefaultError if the annotation type does not exist or if there is ambiguity (multiple sets for the same type)

#### defaultannotatortype (annotationtype, set=None)

Obtain the default annotator type for the specified annotation type and set.

#### **Parameters**

• annotationtype — The type of annotation, this is conveyed by passing the corresponding annotation class (such as *PosAnnotation* for example), or a member of AnnotationType, such as AnnotationType.POS.

• **set** (str) – the set, should formally be a URL pointing to the set definition

Returns AnnotatorType.AUTO or AnnotatorType.MANUAL

**Raises** NoDefaultError if the annotation type does not exist or if there is ambiguity (multiple sets for the same type)

#### **defaultdatetime** (annotationtype, set=None)

Obtain the default datetime for the specified annotation type and set.

#### **Parameters**

- annotationtype The type of annotation, this is conveyed by passing the corresponding annotation class (such as *PosAnnotation* for example), or a member of AnnotationType, such as AnnotationType.POS.
- **set** (str) the set, should formally be a URL pointing to the set definition

**Returns** the set (str)

**Raises** NoDefaultError if the annotation type does not exist or if there is ambiguity (multiple sets for the same type)

#### defaultset (annotationtype)

Obtain the default set for the specified annotation type.

**Parameters annotationtype** – The type of annotation, this is conveyed by passing the corresponding annotation class (such as *PosAnnotation* for example), or a member of AnnotationType, such as AnnotationType.POS.

**Returns** the set (str)

**Raises** NoDefaultError if the annotation type does not exist or if there is ambiguity (multiple sets for the same type)

```
findwords (*args, **kwargs)
```

#### items()

Returns a depth-first flat list of all items in the document

#### json()

Serialise the document to a dict ready for serialisation to JSON.

Example:

```
import json
jsondoc = json.dumps(doc.json())
```

#### jsondeclarations()

Return all declarations in a form ready to be serialised to JSON.

Returns list of dict

#### language (value=None)

No arguments: Get the document's language (ISO-639-3) from metadata Argument: Set the document's language (ISO-639-3) in metadata

#### license(value=None)

No arguments: Get the document's license from metadata Argument: Set the document's license in metadata

#### load(filename)

Load a FoLiA XML file.

**Argument:** filename (str): The file to load

```
paragraphs (index=None)
```

Return a generator of all paragraphs found in the document.

If an index is specified, return the n'th paragraph only (starting at 0)

#### parsemetadata(node)

Internal method to parse metadata

#### parsesubmetadata(node)

#### parsexml (node, ParentClass=None)

Internal method.

This is the main XML parser, will invoke class-specific XML parsers.

#### parsexmldeclarations (node)

Internal method to parse XML declarations

#### pendingvalidation (warnonly=None)

Perform any pending validations

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

#### publisher (value=None)

No arguments: Get the document's publisher from metadata Argument: Set the document's publisher in metadata

#### save (filename=None)

Save the document to file.

**Parameters filename** (\*) – The filename to save to. If not set (None, default), saves to the same file as loaded from.

```
select (Class, set=None, recursive=True, ignore=True)
```

See AbstractElement.select()

#### sentences (index=None)

Return a generator of all sentence found in the document. Except for sentences in quotes.

If an index is specified, return the n'th sentence only (starting at 0)

#### setimdi(node)

**OBSOLETE** 

#### text (cls='current', retaintokenisation=False)

Returns the text of the entire document (returns a unicode instance)

#### See also:

```
AbstractElement.text()
```

#### title (value=None)

Get or set the document's title from/in the metadata

No arguments: Get the document's title from metadata Argument: Set the document's title in metadata

#### unalias (annotationtype, alias)

Return the set for an alias (if applicable, raises an exception otherwise)

```
words (index=None)
```

Return a generator of all active words found in the document. Does not descend into annotation layers, alternatives, originals, suggestions.

If an index is specified, return the n'th word only (starting at 0)

#### **xm1**()

Serialise the document to XML.

**Returns** lxml.etree.Element

#### See also:

```
Document.xmlstring()
```

#### xmldeclarations()

Internal method to generate XML nodes for all declarations

#### xmlmetadata()

Internal method to serialize metadata to XML

#### xmlstring()

Return the XML representation of the document as a string.

#### xpath (query)

Run Xpath expression and parse the resulting elements. Don't forget to use the FoLiA namesapace in your expressions, using folia: or the short form f:

To read a document from file, instantiate a document as follows:

```
doc = folia.Document(file="/path/to/document.xml")
```

This returned *Document* instance holds the entire document in memory. Note that for large FoLiA documents this may consume quite some memory! If you happened to already have the document content in a string, you can load as follows:

```
doc = folia.Document(string="<FoLiA ...")</pre>
```

Once you have loaded a document, all data is available for you to read and manipulate as you see fit. We will first illustrate some simple use cases:

To save a document back to the file it was loaded from, we do:

```
doc.save()
```

Or we can specify a specific filename:

```
doc.save("/tmp/document.xml")
```

**Note:** Any content that is in a different XML namespace than the FoLiA namespaces or other supported namespaces (XML, Xlink), will be ignored upon loading and lost when saving.

### 4.1.2 Printing text

You may want to simply print all (plain) text contained in the document, which is as easy as:

```
print(doc)
```

Obtaining the text as a string is done by invoking the document's Document.text () method:

```
text = doc.text()
```

Or alternatively as follows:

```
text = str(doc)
```

For any subelement of the document, you can obtain its text in the same fashion as well, by calling its AbstractElement.text() method or by using str(), the only difference is that the former allows for extensive fine tuning using various extra parameters (See AbstractElement.text()).

Note: In Python 2, both str() as well as unicode() return a unicode instance. You may need to append . encode('utf-8') for proper output.

#### 4.1.3 Index

A document instance has an **index** which you can use to grab any of its elements by ID. Querying using the index proceeds similar to using a python dictionary:

```
word = doc['example.p.3.s.5.w.1']
print(word)
```

**Note:** Python 2 users will have to do print word.text().encode('utf-8') instead, to ensure non-ascii characters are printed properly.

IDs are unique in the entire document, and preferably even beyond.

#### 4.1.4 Elements

All FoLiA elements are derived from AbstractElement and offer an identical interface. To quickly check whether you are dealing with a FoLiA element you can therefore always do the following:

```
isinstance(word, folia.AbstractElement)
```

This abstract base element is never instantiated directly. The FoLiA paradigm derives several more abstract base classes which may implement some additional methods or overload some of the original ones:

AbstractElement	Abstract base class from which all FoLiA elements are derived.
AbstractStructureElement	Abstract element, all structure elements inherit from this
	class.
AllowTokenAnnotation	Elements that allow token annotation (including ex-
	tended annotation) must inherit from this class
AbstractSpanAnnotation	Abstract element, all span annotation elements are de-
	rived from this class
AbstractTokenAnnotation	Abstract element, all token annotation elements are de-
	rived from this class
	0 11 1

Continued on next page

T 1 1 4		•	•	
10010/	AANTINIIAA	trom	nravialia	n - n - n
101115 4 -	COMMINICA	11()111	IN EAIMS	nauc
	continued	•	p. 0 0 0. 0	P~9~

AbstractAnnotationLayer	Annotation layers for Span Annotation are derived from
	this abstract base class
AbstractTextMarkup	Abstract class for text markup elements, elements that
	appear with the TextContent (t) element.

#### pynlpl.formats.folia.AbstractElement

Abstract base class from which all FoLiA elements are derived.

This class implements many generic methods that are available on all FoLiA elements.

To see if an element is a FoLiA element, as opposed to any other python object, do:

isinstance(x, AbstractElement)

Generic FoLiA attributes can be accessed on all instances derived from this class:

- element .id (str) The unique identifier of the element
- element.set (str) The set the element pertains to.
- element.cls (str) The assigned class, i.e. the actual value of the annotation, defined in the set. Classes correspond with tagsets in this case of many annotation types. Note that since *class* is already a reserved keyword in python, the library consistently uses cls everywhere.
- element .annotator (str) The name or ID of the annotator who added/modified this element
- element.annotatortype The type of annotator, can be either folia.AnnotatorType. MANUAL or folia.AnnotatorType.AUTO
- element.confidence (float) A confidence value expressing
- element . datetime (datetime.datetime) The date and time when the element was added/modified.
- element.n (str) An ordinal label, used for instance in enumerated list contexts, numbered sections, etc..

The following generic attributes are specific to a speech context:

- element.src (str) A URL or filename referring the an audio or video file containing the speech. Access this attribute using the element.speaker\_src() method, as it is inheritable from ancestors.
- element.speaker (str) The name of ID of the speaker. Access this attribute using the element. speech\_speaker() method, as it is inheritable from ancestors.
- element.begintime (4-tuple) The time in the above source fragment when the phonetic content of this element starts, this is a (hours, minutes, seconds, milliseconds) tuple.
- element . endtime (4-tuple) The time in the above source fragment when the phonetic content of this element ends, this is a (hours, minutes, seconds, milliseconds) tuple.

Not all attributes are allowed, unset or unavailable attributes will always default to None.

**Note:** This class should never be instantiated directly, as it is abstract!

See also:

AbstractElement.\_\_init\_\_()

### **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
	Continued on next page

Table 5 – continued from previous page
----------------------------------------

	ed from previous page
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
_	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
5 (t - 1 mars - 1 mar	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
Top Taco (vinia, ango, invargo)	places any existing child element of the same type
	and set.
resolveword(id)	and see
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
setparents()	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
speecii_speakei()	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
speecii_sic()	associated with the element.
stricttext([cls])	Alias for text() with strict=True
	Get the text associated with this element (of the spec-
text([cls, retaintokenisation,])	
tout cont ont ([els correctionhandling])	ified class)  Get the text content explicitly associated with this
textcontent([cls, correctionhandling])	element (of the specified class).
torresolidation([rumanle])	
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
7/[ // 1	the children.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

#### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL ATTRIBS = None
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = None
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
___init___(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
```

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

Returns bool

```
Raises ValueError
```

#### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

#### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

#### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

#### deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

#### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### **Returns** bool

```
hastext (cls='current', strict=True, correctionhandling=1)
```

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

#### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

**phon** (*cls='current'*, *previousdelimiter="*, *strict=False*, *correctionhandling=1*)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- retaintokenisation (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

#### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

#### Example:

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

#### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

text() phoncontent() phon()

```
textvalidation(warnonly=None)
```

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

See also:

AbstractElement.xmlstring() - for direct string output

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.AbstractStructureElement

Abstract element, all structure elements inherit from this class. Never instantiated directly.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
gettextdelimiter([retaintokenisation])	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
hastext(ICIS SITICI COTTECTIONNANGIINGI)	

4.1. Reading FoLiA

Table 6 – continue	ed from previous page
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
(5 - 1)	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> un-
7. 6	der this element, does not include alternative layers
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
oniginal+ou+([ala])	Scope.  Alias for ratriaving the original uncorrect taxt
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
<pre>parsexml(node, doc, **kwargs)</pre>	cursively) under this element.  Internal class method used for turning an XML ele-
par sexur (node, doc, "kwargs)	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
phon([cis, previousdeminter, strict,])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
phoneoneene ([eis, correction and mg])	this element (of the specified class).
postappend()	This method will be called after an element is added
postappena()	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
[FIGURE ([Class, SeePel)	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
, , , , , , , , , , , , , , , , , , ,	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the specified class)
	Continued on next page

Table 6 – continued from previous page

<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = '\n\n'
XLINK = False
XMLTAG = None
```

4.1. Reading FoLiA

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

# ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

#### annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

# annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

# **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

# See also:

AbstractElement.select()

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

# Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.AllowTokenAnnotation

```
class pynlpl.formats.folia.AllowTokenAnnotation
    Bases: pynlpl.formats.folia.AllowCorrections
```

Elements that allow token annotation (including extended annotation) must inherit from this class

# **Method Summary**

alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also
	by set.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
str	Return str(self).

# **Method Details**

\_\_\_init\_\_\_()

Initialize self. See help(type(self)) for accurate signature.

### alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

# **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

### annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

# See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

#### See also:

AbstractElement.select()

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
hasannotation(Class, set=None)
```

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

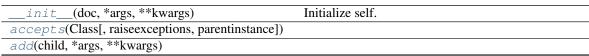
```
__str__()
```

Return str(self).

# pynlpl.formats.folia.AbstractSpanAnnotation

Abstract element, all span annotation elements are derived from this class

## **Method Summary**



Continued on next page

Table 8 – continu	led from previous page
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
<i>C</i> /	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
, , , , , , , , , , , , , , , , , , , ,	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	<u> </u>
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
, <u>,</u> , , , , , , , , , , , , , , , , ,	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
V	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
7 6	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
, L/1/ T-1/	Continued on next page

3 – continued		

	ed from previous page
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
- · · · · · · · · · · · · · · · · · · ·	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
, , ,	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
setspan(*args)	Sets the span of the span element anew, erases all
	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
( )	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
\L	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
wrefs([index, recurse])	Returns a list of word references, these can be Words
- (L,1/	but also Morphemes or Phonemes.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter ()	Iterate over all children of this element.
	Continued on next page
	Continued on next page

Table 8 – continued from previous page

len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = None
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
```

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

# **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

#### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

# addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation (type, set=None)

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

# annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*) -
- set The set you want to retrieve (\*)-

# Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

# copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

### **Parameters**

• newdoc(Document) - The document the copy should be associated with.

• idsuffix (str or bool) – If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

```
\mathtt{generate\_id}\left(\mathit{cls}\right)
```

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

# Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

```
remove (child)
```

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

## Example:

## setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

## setspan(\*args)

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text. cls='current')
```

Set the text for this element.

#### **Parameters**

- text(str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

#### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls(str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

 normalize\_spaces (bool) – Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

#### toktext (cls='current')

Alias for text() with retaintokenisation=True

## updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### wrefs (index=None, recurse=True)

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children **Return type** str

```
__iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.AbstractTokenAnnotation

Abstract element, all token annotation elements are derived from this class

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
	Continued on next page

4.1. Reading FoLiA

Table 9 – Continu	ed from previous page
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined
and oid no 14 and (falal)	Scope.
<pre>originaltext([cls]) parsexml(node, doc, **kwargs)</pre>	Alias for retrieving the original uncorrect text.
	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but replaces any existing child element of the same type and set.
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the scop.
	Set the text for this element.

Table 9 – continued from previous page

Retrieves the speaker of the audio or video file asso-
ciated with the element.
Retrieves the URL/filename of the audio or video file
associated with the element.
Alias for text() with strict=True
Get the text associated with this element (of the spec-
ified class)
Get the text content explicitly associated with this
element (of the specified class).
Run text validation on this element.
Alias for text() with
retaintokenisation=True
Recompute textual value based on the text content of
the children.
Serialises the FoLiA element and all its contents to
XML.
Serialises this FoLiA element and all its contents to
XML.
Iterate over all children of this element.
Returns the number of child elements under the cur-
rent element.
Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 1
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED\_ATTRIBS = (1,)
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
```

#### XMLTAG = None

# **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

# findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
    node - XML Element (*) -
    doc - Document (*) -
```

**Returns** An instance of the current Class.

```
\textbf{phon} \ (cls='current', previous de limiter=", strict=False, correction handling=1")
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT,

which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

```
phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()
```

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
\verb|select| (Class, set=None, recursive=True, ignore=True, node=None)|
```

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

### Parameters doc (Document) - A document

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

• normalize\_spaces (bool) - Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

#### toktext (cls='current')

Alias for text() with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.AbstractAnnotationLayer

Annotation layers for Span Annotation are derived from this abstract base class

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a single annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
	Continued on next page

Table 10 – continued from previous page

	led from previous page
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec-
- , ,	ified words or morphemes.
generate_id(cls)	*
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
9	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
11100110001011()	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
([])	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
2 (L	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
Total of Canada, acope, reversely	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
par seame (noue, doe, kwaigs)	ment into an instance of the Class.
phon([cls previousdelimiter strict 1)	Get the phonetic representation associated with this
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phon([cls, previousdelimiter, strict,]) phoncontent([cls, correctionhandling])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with
phoncontent([cls, correctionhandling])	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).
	element (of the specified class)  Get the phonetic content explicitly associated with

4.1. Reading FoLiA

Table 10	) – continued	from pr	revious page

Table 10 – continue	d from previous page	
previous([Class, scope])	Returns the previous element, if it is of the specified	
	type and if it does not cross the boundary of the de-	
	fined scope.	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.	
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text()	

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = None

AUTH = True

AUTO_GENERATE_ID = False

OCCURRENCES = 0
```

OCCURRENCES\_PER\_SET = 0

OPTIONAL\_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)

```
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = None
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
             • parent (AbstractElement) – The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
        Raises ValueError
addidsuffix (idsuffix, recursive=True)
    Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
    call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
    Makes sure this element (and all subelements), are properly added to the index.
    Mostly for internal use.
alternatives (Class=None, set=None)
```

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by

4.1. Reading FoLiA

set.

### **Parameters**

- Class The Class you want to retrieve (\*) -
- set The set you want to retrieve (\*) -

**Returns** Generator over Alternative elements

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation (type, set=None)

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

### annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

## **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*)-

# Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

### correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

# count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

## findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

#### See also:

```
Word.findspans()
```

### generate\_id(cls)

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

## **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

### originaltext(cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
\verb|select| (Class, set=None, recursive=True, ignore=True, node=None)|
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy* ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

Example:

```
word.text()
```

Returns The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

**Return type** str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:

__len__()
Returns the number of child elements under the current element.

__str__()
Alias for text()
```

# pynlpl.formats.folia.AbstractTextMarkup

```
class pynlpl.formats.folia.AbstractTextMarkup(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Abstract class for text markup elements, elements that appear with the TextContent (t) element.

Markup elements pertain primarily to styling, but also have other roles.

Iterating over the element of a *TextContent* element will first and foremost produce strings, but also uncover these markup elements when present.

# **Method Summary**

init(doc, *args, **kwargs)	See AbstractElementinit(), text is
(doc, args, kwargs)	passed as a string in *args.
accepts(Class[, raiseexceptions, parentinstance])	pussed us a string in Adags.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addab 1 e(pareint, set, raiscenceptions)	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
	Continued on next page

Table 11 – continued from previous page

	ied from previous page
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
V V	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
items([founditems])	Returns a depth-first flat list of all items below this
reems([roundrems])	element (not limited to AbstractElement)
	·
json([attribs, recurse, ignorelist])	See AbstractElement.json()
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
<pre>next([Class, scope, reverse])</pre>	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
(7.1.3)	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
rozamy ([meradoomaron, onwadances,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace(cinia, args, kwargs)	places any existing child element of the same type
	and set.
resolve()	und sot.
resolveword(id)	
	Datume the might context for an alamant as a 1's
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text)	Sets the text content of the markup element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
	Continued on next page

Table 11 – continued from previous page

stricttext([cls])	Alias for text() with strict=True	
t ext([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text ()	

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractTextMarkup'>, <class 'pynlpl.formats.folia.AbstractTex
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = True
TEXTDELIMITER = ''
XLINK = True
XMLTAG = None
```

## **Method Details**

```
__init__ (doc, *args, **kwargs)
See AbstractElement.__init__ (), text is passed as a string in *args.
__init__ (doc, *args, **kwargs)
See AbstractElement.__init__ (), text is passed as a string in *args.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

Returns int

## deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

## description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

#### findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# $\verb"getindex" (child, recursive=True, ignore=True)"$

Get the index at which an element occurs, recursive by default!

Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

**hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
See AbstractElement.json()
```

```
leftcontext (size, placeholder=None, scope=None)
```

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

## originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

# remove(child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolve()
```

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by <code>copy()</code>

```
settext(text)
```

Sets the text content of the markup element.

```
Parameters text (str) -
```

```
speech_speaker()
```

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

• correctionhandling – Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

### Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# 4.1.5 Obtaining list of elements

The aforementioned index is useful only if you know the ID of the element. This if often not the case, and you will want to iterate through the hierarchy of elements through different means.

If you want to iterate over all of the child elements of a certain element, regardless of what type they are, you can simply do so as follows:

```
for subelement in element:
   if isinstance(subelement, folia.Sentence):
        print("this is a sentence")
   else:
        print("this is something else")
```

If applied recursively this allows you to traverse the entire element tree, there are however specialised methods available that do this for you.

## 4.1.6 Select method

There is a generic method <code>AbstractElement.select()</code> available on all elements to select child elements of any desired class. This method is by default applied recursively for most element types:

```
sentence = doc['example.p.3.s.5.w.1']
words = sentence.select(folia.Word)
for word in words:
    print(word)
```

The AbstractElement.select() method has a sibling AbstractElement.count(), invoked with the same arguments, which simply counts how many items it finds, without actually returning them:

```
word = sentence.count(folia.Word)
```

**Note:** The select () method and similar high-level methods derived from it, are generators. This implies that the results of the selection are returned one by one in the iteration, as opposed to all stored in memory. This also implies that you can only iterate over it once, we can not do another iteration over the words variable in the above example, unless we reinvoke the select () method to get a new generator. Likewise, we can not do len (words), but have to use the count () method instead.

If you want to have all results in memory in a list, you can simply do the following:

```
words = list(sentence.select(folia.Word))
```

The select method is by default recursive, set the third argument to False to make it non-recursive. The second argument can be used for restricting matches to a specific set, a tuple of classes. The recursion will not go into any *non-authoritative* elements such as alternatives, originals of corrections.

# 4.1.7 Selection Shortcuts

There are various shortcut methods for select ().

For example, you can iterate over all words in the document using <code>Document.words()</code>, or all words under any structural element using <code>AbstractStructureElement.words()</code>:

```
for word in doc.words():
    print(word)
```

That however gives you one big iteration of words without boundaries. You may more likely want to seek words within sentences, provided the document distinguishes sentences. So we first iterate over all sentences using <code>Document.sentences()</code> and then over the words therein using <code>AbstractStructureElement.words()</code>:

```
for sentence in doc.sentences():
    for word in sentence.words():
        print(word)
```

Or including paragraphs, assuming the document has them:

```
for paragraph in doc.paragraphs():
    for sentence in paragraph.sentences():
        for word in sentence.words():
            print(word)
```

**Warning:** Do be aware that such constructions make presumptions about the structure of the FoLiA document that may not always apply!

All of these shortcut methods also take an index parameter to quickly select a specific item in the sequence:

```
word = sentence.words(3) #retrieves the fourth word
```

# 4.1.8 Navigating a document

The AbstractElement.select() method is your main tool for descending downwards in the document tree. There are occassions, however, when you want go upwards or sideways. The AbstractElement.next() and AbstractElement.previous() methods can be used for sideway navigation, they return the next or previous element, respectively:

```
nextelement = element.next()
previouselement = element.previous()
```

You can explicitly filter by passing an element type:

```
nextword = word.next(folia.Word)
```

By default, the search is constrained not to cross certain boundaries, such as sentences and paragraphs. You can do so explicitly as well by passing a list of constraints:

```
nextword = word.next(folia.Word, [folia.Sentence])
```

If you do not want any constraints, pass None:

```
nextword = word.next(folia.Word, None)
```

These methods will return None if no next/previous element was found (of the specified type).

Each element has a parent attribute that links it to its parent:

```
sentence = word.parent
```

Only for the top-level element (Text or Speech), the parent is None. There is also the method AbstractElement.ancestors() to iterate over all ancestors, ordered from most immediate to most distant ancestor:

```
for ancestor in element.ancestors():
    print(type(ancestor))
```

If you are looking for ancestors of a specific type, you can pass it as an argument:

```
for ancestor in element.ancestors(folia.Division):
    print(type(ancestor))
```

If only a single ancestor is desired, use the <code>AbstractElement.ancestor()</code> method instead, unlike the generator version <code>AbstractElement.ancestors()</code>, it will raise a <code>NoSuchAnnotation</code> exception if the ancestor was not found:

```
paragraph = word.ancestor(folia.Paragraph)
```

# 4.1.9 Structure Annotation Types

The FoLiA library discerns various Python classes for structure annotation, all are subclasses of AbstractStructureElement, which in turn is a subclass of AbstractElement. We list the classes for structure annotation along with the FoLiA XML tag. Sets and classes can be associated with most of these elements to make them more specific, these are never prescribed by FoLiA. The list of classes is as follows:

Cell	A cell in a Row in a Table
Definition	Element used in <i>Entry</i> for the portion that provides a
	definition for the entry.
Division	Structure element representing some kind of division.
Entry	Represents an entry in a glossary/lexicon/dictionary.
Event	Structural element representing events, often used in
	new media contexts for things such as tweets, chat mes-
	sages and forum posts.
Example	Element that provides an example.
Figure	Element for the representation of a graphical figure.
Gap	Gap element, represents skipped portions of the text.
Head	Head element; a structure element that acts as the
	header/title of a Division.
Linebreak	Line break element, signals a line break.
List	Element for enumeration/itemisation.
ListItem	Single element in a List.
Note	Element used for notes, such as footnotes or warnings
	or notice blocks.
Paragraph	Paragraph element.
Part	Generic structure element used to mark a part inside an-
	other block.
Quote	Quote: a structure element.
Reference	A structural element that denotes a reference, internal or
	external.
Row	A row in a Table
Sentence	Sentence element.
Table	A table consisting of Row elements that in turn consist
	of Cell elements
Term	A term, often used in contect of Entry
TableHead	Encapsulated the header of a table, contains Cell ele-
	ments
Text	A full text.
Whitespace	Whitespace element, signals a vertical whitespace
	Continued on next page

Continued on next page

# Table 12 – continued from previous page

	<u> </u>	
Word	Word (aka token) element.	

# pynlpl.formats.folia.Cell

class pynlpl.formats.folia.Cell(doc, \*args, \*\*kwargs)

Bases: pynlpl.formats.folia.AbstractStructureElement

A cell in a Row in a Table

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
	Continued on next page

Table 13 – continued from previous page

Table 13 – continue	ed from previous page
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	wise it retains from
items([founditems])	Returns a depth-first flat list of <i>all</i> items below this
T Cemes([Iounditems])	element (not limited to AbstractElement)
icon([attribe requires ignoralist])	Serialises the FoLiA element and all its contents to a
<pre>json([attribs, recurse, ignorelist])</pre>	
Tarrage (Commototiontyma ==41)	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> under this algorithm and the state of the s
	der this element, does not include alternative layers
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
- -	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
, , ,	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveworduu	
resolveword(id)	Returns the right context for an element, as a list
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
rightcontext(size[, placeholder, scope]) select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
rightcontext(size[, placeholder, scope])	Select child elements of the specified class.  Returns a generator of Sentence elements found (re-
rightcontext(size[, placeholder, scope]) select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.

4.1. Reading FoLiA

Table 13 – continued from previous page

Table 13 – continued from previous page		
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
words([index])	Returns a generator of Word elements found (recur-	
	sively) under this element.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len ()	Returns the number of child elements under the cur-	
()	Returns the number of child elements under the cur-	
	rent element.	

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Cell'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

# Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- Class (class) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (str) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

## annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

# **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

### See also:

```
AbstractElement.select()
```

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

### hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

## **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

# hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling – Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

# originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correction handling = Correction Handling . ORIGINAL

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if

you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & orig-class = None) \\ \end{tabular}$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

## remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the replaced element will be made into an alternative. Simply use AbstractElement.append() if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

# ${\tt resolveword}\,(id)$

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

#### Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

 $\label{text} \begin{tabular}{ll} \textbf{text} (cls='current', retain to ken is at ion=False, previous delimiter=", strict=False, correction handling=1, normalize\_spaces=False) \\ \end{tabular}$ 

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if

you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

```
Alias for text () with retain token is at ion=True
```

## updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

## See also:

```
AbstractElement.xmlstring() - for direct string output
```

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.Definition

class pynlpl.formats.folia.Definition(doc, \*args, \*\*kwargs)
 Bases: pynlpl.formats.folia.AbstractStructureElement

Element used in *Entry* for the portion that provides a definition for the entry.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
(partial)	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
(L	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
	Continued on next page

4.1. Reading FoLiA

Table 14 – continued from previous page

Returns the text delimiter for this class.		ed from previous page
tation exists, and if so, how many.  hasannotationlayer([annotationtype, set])  hasannotationlayer([cls, strict, correctionhandling])  noes this element have phonetic content (of the specified class)  nocrrection()  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  seinment (not limited to AbstractElement)  serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns a list of annotation layers found directly under this element, does not include alternative layers (lannotationtype, set])  Returns the list of annotation layers found directly under this element, does not include alternative layers and if it does not cross the boundary of the defined scope.  originaltext([cls])  paragraphs([index])  paragraphs([index])  paragraphs([index])  phon([cls, previousdelimiter, strict,])  phonocontent([cls, correctionhandling])  phonocontent([cls, correctionhandling])  phonocontent([cls, correctionhandling])  postappend()  This method will be called after an element is and another and does some checks.  Previous([Class, scope])  Returns a RelaxNG definition for this element is an XML element (of the specified class)  Returns a penerator of Paragraph elements found (recursively) under this element.  Returns a generator of paragraph elements found (recursively) under this element.  Returns a generator of Paragraph elements found (recursively) under this element (as an XML element (of the specified class).  Returns a generator of paragraph element is added to another and does some checks.  Returns a RelaxNG definition for this element (as an XML element (with the called after an element is an XML element (with the sent of the specified class).  Returns a RelaxNG definition for this element (as an XML element (with this element (as	<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
Does the specified annotation layer exist?	hasannotation(Class[, set])	
Does this element have phonetic content (of the specified class)   Incorrection()		
ified class)    hastext([cls, strict, correctionhandling])   Does this element have text (of the specified class)     incorrection()   Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None     insert(index, child, *args, **kwargs)   Returns a depth-first flat list of all items below this element (not limited to AbstractElement)     json([attribs, recurse, ignorelist])   Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.     layers([annotationtype, set])   Returns a list of annotation layers found directly under this element, does not include alternative layers     leftcontext(size], placeholder, scope])   Returns the left context for an element, as a list.     next([Class, scope, reverse])   Returns the left context for an element, as a list.     paragraphs([index])   Alias for retrieving the original uncorrect text.     paragraphs([index])   Returns a generator of Paragraph elements found (recursively) under this element.     paragraphs([index])   Returns a generator of Paragraph elements found (recursively) under this element.     paragraphs([index])   Returns a generator of Paragraph elements found (recursively) under this element.     paragraphs([index])   Returns a generator of Paragraph elements found (recursively) under this element.     phon([cls, previousdelimiter, strict,])   Get the phonetic content explicitly associated with this element (of the specified class)     postappend()   This method will be called after an element is added to another and does some checks.     previous([Class, scope])   Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.     remove(child)   Returns the previous element, if it is of the specified class)     remove(child)   Returns the previous element (as an XML element (as an XML element (as an XML element (as an yexisting child element of the same type and set.     reso		
Incorrection()   Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None     Insert (index, child, *args, **kwargs)     Items([founditems])   Returns a depth-first flat list of all items below this element (not limited to AbstractElement)     Json([attribs, recurse, ignorelist])   Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.     Iayers([annotationtype, set])   Returns a list of annotation layers found directly under this element, does not include alternative layers	hasphon([cls, strict, correctionhandling])	
the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  fson([attribs, recurse, ignorelist])  fson([attribs, clement, and il its of hesperialistation to JSON)  Returns a list of annotation layers found directly under this clement, does not include alternative layers  and if it does not cross the boundary of the defined scope.  featurns a generator of Paragraph elements found (recursively) under this element.  for the violation of paragraph elements found (recursively) under this element.  for the phonetic order explicitly associated with this element (of the specified class)  phone(attrib, and in the previous element, in it is of the specified vipe and if it does not cross the boundary of the defined scope.  frelaxng([includechildren, extraattribs,])  fremove(child)  replace(child, *args, **kwargs)  fremove(child)  rep	hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
insert(index, child, *args, **kwargs)           items([founditems])         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           layers([annotationtype, set])         Returns a list of annotation layers found directly under this element, does not include alternative lader this element, does not include alternative lader this element, does not cross the boundary of the defined scope.           next([Class, scope, reverse])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           original text([cls])         Alias for retrieving the original uncorrect text.           paragraphs([index])         Returns a generator of Paragraph elements found (recursively) under this element.           phon([cls, previousdelimiter, strict,])         Returns a generation associated with this element (of the specified class)           phon([cls, previousdelimiter, strict,])         Get the phonetic content explicitly associated with this element (of the specified class).           phon([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           previous([Class, scope])         Returns the previous element, if it is of the specified class and if it does not cross t	incorrection()	Is this element part of a correction? If it is, it returns
Returns a depth-first flat list of all items below this element (not limited to AbstractElement)		
element (not limited to AbstractElement)    Json([attribs, recurse, ignorelist])   Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.   Layers([annotationtype, set])   Returns a list of annotation layers found directly under this element, does not include alternative layers	<pre>insert(index, child, *args, **kwargs)</pre>	
Python dictionary suitable for serialisation to JSON.	<pre>items([founditems])</pre>	
Returns a list of annotation layers found directly under this element, does not include alternative layers	json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
der this element, does not include alternative layers		<u> </u>
leftcontext(size[, placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           originaltext([cls])         Alias for retrieving the original uncorrect text.           Returns a generator of Paragraph elements found (recursively) under this element.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class).           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxing([includechildren, extraattribs,])         Returns the previous element, if it is of the specified class).           previous([Class, scope])         Returns the previous element, if it is of the specified class on the case and the specified class.           previous([Class, scope])         Returns the previous element, if it is of the specified class on the specified class.           remove(child)         Removes the chi	layers([annotationtype, set])	
next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           originaltext([cls])         Alias for retrieving the original uncorrect text.           paragraphs([index])         Returns a generator of Paragraph elements found (recursively) under this element.           phon([cls, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns a Previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxing([includechildren, extraattribs,])         Returns a Relaxing definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         Returns the right context for an element, as a list.           select(Class], set, recursive, ignore, node])         Returns the right context for an element, as a list.           select(clidex])         Returns a generator of Sentence elements found (recursivel		
and if it does not cross the boundary of the defined scope.    Paragraphs([index])		
originaltext([cls])         Alias for retrieving the original uncorrect text.           paragraphs([index])         Returns a generator of Paragraph elements found (recursively) under this element.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         resolveword(id)           rightcontext(size[, placeholder, scope])         Returns the right context for an element, as a list.           select(Class], set, recursive, ignore, node])         Select child elements of the specified class.           s	next([Class, scope, reverse])	
originaltext([cls])         Alias for retrieving the original uncorrect text.           paragraphs([index])         Returns a generator of Paragraph elements found (recursively) under this element.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict, ])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         Returns the right context for an element, as a list.           select(Class[, set, recursive, ignore, node])         Select child elements of the specified class.           sentences([index])         Returns a generator of Sentence elements found (recursively) under thi		•
paragraphs([index])         Returns a generator of Paragraph elements found (recursively) under this element.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         Returns the right context for an element, as a list.           select(Class], set, recursive, ignore, node])         Select child elements of the specified class.           sentences([index])         Returns a generator of Sentence elements found (recursively) under this element           setdoc(newdoc)         Set a different document.	(5.1.2)	
cursively) under this element.  parsexml (node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Remove the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(Class, set, recursive, ignore, node])  select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  setdoc(newdoc)  Select child element with this element.  setdocument(doc)  Associate a document with this element.  setdocument (set)  Settext(text[, cls])  Set the text for this element.		
ment into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(Class[, set, recursive, ignore, node])  select child elements of the specified class.  sentences([index])  Returns a generator of Sentence elements found (recursively) under this element  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setdocuments within the scop.  settext(text[, cls])  Set the text for this element.		cursively) under this element.
phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         Returns the right context for an element, as a list.           select(Class[, set, recursive, ignore, node])         Select child elements of the specified class.           sentences([index])         Returns a generator of Sentence elements found (recursively) under this element           setdoc(newdoc)         Set a different document.           setdocument(doc)         Associate a document with this element.           setparents()         Correct all parent relations for elements within the scop.           settext(text[, cls])         Set the text for this element.	<pre>parsexml(node, doc, **kwargs)</pre>	•
element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(Class[, set, recursive, ignore, node])  select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.		
phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         Returns the right context for an element, as a list.           select(Class[, set, recursive, ignore, node])         Select child elements of the specified class.           sentences([index])         Returns a generator of Sentence elements found (recursively) under this element           setdoc(newdoc)         Set a different document.           setdocument(doc)         Associate a document with this element.           setparents()         Correct all parent relations for elements within the scop.           settext(text[, cls])         Set the text for this element.	phon([cls, previousdelimiter, strict,])	
this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element ( xml.etree ) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  Select child elements of the specified class.  sentences([index])  Returns a generator of Sentence elements found (recursively) under this element  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.	(5.1	
This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(Class[, set, recursive, ignore, node])  select child elements of the specified class.  select(Class[, set, recursive, ignore, node])  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.	phoncontent([cls, correctionhandling])	· · · · · · · · · · · · · · · · · · ·
to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(Class[, set, recursive, ignore, node])  select(clindex])  Returns the right context for an element, as a list.  select(class[, set, recursive, ignore, node])  select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  set document(doc)  Associate a document with this element.  set parents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.		
type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(Class[, set, recursive, ignore, node])  sentences([index])  Returns a generator of Sentence elements found (recursively) under this element  setdoc(newdoc)  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.		to another and does some checks.
relaxng([includechildren, extraattribs,])Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)Returns the right context for an element, as a list.rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.sentences([index])Returns a generator of Sentence elements found (recursively) under this elementsetdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.	previous([Class, scope])	type and if it does not cross the boundary of the de-
remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.sentences([index])Returns a generator of Sentence elements found (recursively) under this elementsetdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.		
replace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)Returns the right context for an element, as a list.rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.sentences([index])Returns a generator of Sentence elements found (recursively) under this elementsetdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.		XML element (lxml.etree) rather than a string)
places any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope]) Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node]) Select child elements of the specified class.  sentences([index]) Returns a generator of Sentence elements found (recursively) under this element  setdoc(newdoc) Set a different document.  setdocument(doc) Associate a document with this element.  setparents() Correct all parent relations for elements within the scop.  settext(text[, cls]) Set the text for this element.		
and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  sentences([index])  Returns the right context for an element, as a list.  Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.	replace(child, *args, **kwargs)	= =
resolveword(id)rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.sentences([index])Returns a generator of Sentence elements found (recursively) under this elementsetdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.		
rightcontext(size[, placeholder, scope])       Returns the right context for an element, as a list.         select(Class[, set, recursive, ignore, node])       Select child elements of the specified class.         sentences([index])       Returns a generator of Sentence elements found (recursively) under this element         setdoc(newdoc)       Set a different document.         setdocument(doc)       Associate a document with this element.         setparents()       Correct all parent relations for elements within the scop.         settext(text[, cls])       Set the text for this element.	- (1.1)	and set.
select(Class[, set, recursive, ignore, node])       Select child elements of the specified class.         sentences([index])       Returns a generator of Sentence elements found (recursively) under this element         setdoc(newdoc)       Set a different document.         setdocument(doc)       Associate a document with this element.         setparents()       Correct all parent relations for elements within the scop.         settext(text[, cls])       Set the text for this element.		
sentences([index])       Returns a generator of Sentence elements found (recursively) under this element         setdoc(newdoc)       Set a different document.         setdocument(doc)       Associate a document with this element.         setparents()       Correct all parent relations for elements within the scop.         settext(text[, cls])       Set the text for this element.		<del>-</del>
setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.		
setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.		cursively) under this element
setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.		
scop.       settext(text[, cls])       Set the text for this element.	<u> </u>	
settext(text[, cls]) Set the text for this element.	setparents()	-
·	a a + + a v + (tavt[ ala])	
	Sectext(text[, cis])	

Continued on next page

Table 14 – continued from previous page

Table 11 Continue	a nom providuo pago
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 39
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Definition'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

## **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

# Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

## **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

**Yields** Alternative elements

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

# See also:

```
AllowTokenAnnotation.annotations() AbstractElement.select()
```

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

Yields Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

# Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy()* on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

# Returns int

# ${\tt deepvalidation}\,(\,)$

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

## generate\_id(cls)

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is <code>CorrectionHandling.CURRENT</code>, which will retrieve the corrected/current phonetic content. You can set this to <code>CorrectionHandling.ORIGINAL</code> if you want the phonetic content prior to correction, and <code>CorrectionHandling.EITHER</code> if you don't care.

#### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you

want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

## Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element ( *) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

# **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy* ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Division

```
class pynlpl.formats.folia.Division(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Structure element representing some kind of division. Divisions may be nested at will, and may include almost all kinds of other structure elements.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	minute seri.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addas 1 s (paramet, see, raiseer, see parents)	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
<pre>annotation(type[, set])</pre>	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	~
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
gettextdelimiter([retaintokenisation])	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
	Continued on next page

Table 15 – continued from previous page

	ed from previous page
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
head()	\ 1
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> under this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (recursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but replaces any existing child element of the same type and set.
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (recursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the scop.
settext(text[, cls])	Set the text for this element.

Table 15 – continued from previous page

speech src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
, 1,	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 2
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Division'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = '\n\n'
```

```
XLINK = False
XMLTAG = 'div'
```

# **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

```
alternatives (Class=None, set=None)
```

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

## **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation(type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

# Raises

• AllowTokenAnnotation.annotations()

• NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

## generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### head()

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

Returns An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

## See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

```
textvalidation (warnonly=None)
```

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
___len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Entry

Represents an entry in a glossary/lexicon/dictionary.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
<pre>annotation(type[, set])</pre>	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
nasannotationlayer([annotationtype, set])	Continued on next pag

4.1. Reading FoLiA

hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
masprom(reis, surec, correctionmanding)	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
(5	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> under this element does not include alternative layers
lest content (size[ mlessholder seems])	der this element, does not include alternative layers  Returns the left context for an element, as a list.
<pre>leftcontext(size[, placeholder, scope]) next([Class, scope, reverse])</pre>	
meac([Class, scope, levelse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
[	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
(101)	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
relaxng([includechildren, extraattribs,])	fined scope.  Returns a RelaxNG definition for this element (as an
reraxing([includecimaten, extraatmos,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace(emia, args, kwargs)	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
	Correct all parent relations for elements within the
setparents()	
	scop.
settext(text[, cls])	Set the text for this element.
	Set the text for this element.  Retrieves the speaker of the audio or video file asso-
settext(text[, cls])	Set the text for this element.

Continued on next page

associated with the element.

Table 16 – continued from previous page

stricttext([cls])	Alias for text() with strict=True
(E 2)	
t ext([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 37
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Entry'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = '\n\n'
XLINK = False
XMLTAG = 'entry'
```

## **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

## **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

# ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

## Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

## See also:

```
AbstractElement.select()
```

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

### context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

## findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

### hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

- node XML Element (\*) -
- doc Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

## Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Event

Structural element representing events, often used in new media contexts for things such as tweets, chat messages and forum posts.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	Continued on next page

Table 17 – continued from previous page

	led from previous page
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> un-
rayers([aimotationtype, set])	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
nexectemes, scope, reversely	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (recursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
parseami (node, doe, kwargs)	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
prom(tels, previousaemmer, suret, ])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
1 1 1	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (recursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
	Continued on next page

Continued on next page

Table 17 – continued from previous page

speech src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
text([cis, retaintokenisation,])	ified class)
(5.1	
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
-	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 21
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Event'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n'
```

```
XLINK = False
XMLTAG = 'event'
Method Details
```

```
__init__(doc, *args, **kwargs)
Initialize self. See help(type(self)) f
```

Initialize self. See help(type(self)) for accurate signature.

```
__init__(doc, *args, **kwargs)
```

Initialize self. See help(type(self)) for accurate signature.

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

## Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

## Raises

• AllowTokenAnnotation.annotations()

• NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

### correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

## findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

### hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

## Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

### textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Example

```
class pynlpl.formats.folia.Example(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Element that provides an example. Used for instance in the context of Entry

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	tation exists, and it so, now many.

4.1. Reading FoLiA

	ued from previous page
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
([0]	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
us lave efficiency deskildren sytmeetteike 1)	fined scope.  Returns a RelaxNG definition for this element (as an
relaxng([includechildren, extraattribs,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace(clind, largs, lawargs)	places any existing child element of the same type
	and set.
resolveword(id)	and set.
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
sencences([inuex])	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
occparence()	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
or ocon_oreanor()	ciated with the element.
speech src()	Retrieves the URL/filename of the audio or video file
speech_src()	Retrieves the URL/filename of the audio or video file associated with the element.

Continued on next page

Table 18 – continued from previous page

stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	
text([cis, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
-	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 40
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Example'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = '\n\n'
XLINK = False
XMLTAG = 'ex'
```

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

### annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

## Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

## See also:

AbstractElement.select()

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

## findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

### hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

- node XML Element (\*) -
- doc Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Figure

```
class pynlpl.formats.folia.Figure (doc, *args, **kwargs)
Bases: pynlpl.formats.folia.AbstractStructureElement
```

Element for the representation of a graphical figure. Structure element.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
caption()	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-

Table 19 – continued from previous page

	led from previous page
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
items([founditems])	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
	Continued on next page

Continued on next page

Table 19 – continued from previous page

speech src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the spec-	
, 1	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
words([index])	Returns a generator of Word elements found (recur-	
	sively) under this element.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text()	

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 5
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Figure'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n'
```

```
XLINK = False
XMLTAG = 'figure'
```

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

```
alternatives (Class=None, set=None)
```

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
annotation(type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

# Raises

• AllowTokenAnnotation.annotations()

• NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()

caption()

context (size, placeholder=None, scope=None)
    Returns this word in context, {size} words to the left, the current word, and {size} words to the right

copy (newdoc=None, idsuffix=")
```

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

Make a deep copy of this element and all its children.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

# findcorrection handling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

#### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

# **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you

want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text() with correctionhandling=CorrectionHandling.ORIGINAL
```

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

**phon** (*cls='current'*, *previousdelimiter="*, *strict=False*, *correctionhandling=1*)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
```

```
___len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Gap

```
class pynlpl.formats.folia.Gap(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Gap element, represents skipped portions of the text.

Usually contains Content and possibly also a Description element

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
content()	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
	Continued on next page

4.1. Reading FoLiA

Table 20 – continue	d from previous page
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
1 (F	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
([	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
representation, anger, in anger,	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
sceparenes()	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
Specen_Speaker()	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
Specifical Street	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
conduction recaminonementation,])	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
texteontent([cis, correctionnanding])	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for $t \in xt(t)$ with
LUNLEXL([CIS])	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of the children.
vem / (lattribe alamante ekinahildran)	Serialises the FoLiA element and all its contents to
xm1([attribs, elements, skipchildren])	
([muotty, maint])	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
	Continued on next page

Table 20 – continued from previous page

len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 24
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Gap'
OCCURRENCES = 0
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 5, 8, 6, 7, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'gap'
Method Details
___init___(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
```

by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
content()
```

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

# **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### **Returns** str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to

CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) doc Document (\*) -
- Returns An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

# previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- **Class** (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

# remove(child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*) Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

### Example:

### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# $\verb"setdocument" (doc)$

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the

corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

```
Alias for text () with retaintokenisation=True
```

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter___()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

\_\_str\_\_()

Alias for text ()

# pynlpl.formats.folia.Head

class pynlpl.formats.folia.Head(doc, \*args, \*\*kwargs)
 Bases: pynlpl.formats.folia.AbstractStructureElement

Head element; a structure element that acts as the header/title of a Division.

There may be only one per division. Often contains sentences (Sentence) or Words (Word).

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
find correction handling (cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
	Continued on next page

Table 21 – continued from previous page	Table 21	<ul> <li>continued</li> </ul>	from	previous	page
-----------------------------------------	----------	-------------------------------	------	----------	------

	ed from previous page
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
-	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
<u> </u>	Continued on next page

Continued on next page

Table 21 – continued from previous page

Table 21 Continue	a nom providad pago	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
words([index])	Returns a generator of Word elements found (recur-	
	sively) under this element.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text ()	
5t1()	Allas for CEAC ()	

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp ANNOTATIONTYPE = None

AUTH = True

AUTO_GENERATE_ID = True

LABEL = 'Head'

OCCURRENCES = 1

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = True

REQUIRED_ATTRIBS = None

REQUIRED_DATA = None

SETONLY = False

SPEAKABLE = True
```

```
SUBSET = None

TEXTCONTAINER = False

TEXTDELIMITER = '\n\n'

XLINK = False

XMLTAG = 'head'

Method Details
```

classmethod addable(parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

add (child, \*args, \*\*kwargs)

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation(type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

# annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

### Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

AbstractElement.select()

### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# ${\tt deepvalidation}\,(\,)$

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

#### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### Parameters

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you

want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text() with correctionhandling=CorrectionHandling.ORIGINAL
```

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

**phon** (*cls='current'*, *previousdelimiter="*, *strict=False*, *correctionhandling=1*)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike  $t \in xt$  (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the  $T \in xt$  content instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
___len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Linebreak

```
class pynlpl.formats.folia.Linebreak(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractStructureElement, pynlpl.formats.folia.
```

# AbstractTextMarkup

Line break element, signals a line break.

This element acts both as a structure element as well as a text markup element.

# **Method Summary**

init(doc, *args, **kwargs)	See AbstractElementinit(), text is
(doc, 'args, ''kwargs)	passed as a string in *args.
accepts(Class[, raiseexceptions, parentinstance])	passed as a string in Allys.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addable(parent, set, raiseexceptions))	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
addidSullix(lusullix[, lecuisive])	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
addtoffidex([noiccurse])	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
alternatives([Class, stt])	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
anceseor (Classes)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
ances cors ([Class])	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
aimo ed e i ono (Classi, seti)	class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
concerne(onest, processors, scopes)	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
, ,	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
•	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
	Continued on next page

4.1. Reading FoLiA

Table 22 – continued from previous page

	ed from previous page
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
<pre>hasannotationlayer([annotationtype, set])</pre>	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	See AbstractElement.json()
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
parsexml(node, doc)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolve()	
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	
- ~	Correct all parent relations for elements within the
	Correct all parent relations for elements within the scop.
settext(text)	_

Table 22 – continued from previous page

1 1	The state of the s
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 7
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Linebreak'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = True
```

4.1. Reading FoLiA

```
TEXTDELIMITER = ''
XLINK = True
XMLTAG = 'br'
Method Details
```

```
__init___(doc, *args, **kwargs)
See AbstractElement.__init___(), text is passed as a string in *args.
__init___(doc, *args, **kwargs)
See AbstractElement.__init___(), text is passed as a string in *args.
```

 $\verb|classmethod| accepts| (\textit{Class}, \textit{raiseexceptions} = \textit{True}, \textit{parentinstance} = \textit{None})$ 

```
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

## Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

## **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

## annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

## Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

## See also:

```
AllowTokenAnnotation.annotations() AbstractElement.select()
```

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

Yields Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

## Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

## **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you

want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

```
See AbstractElement. json ()
```

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

```
leftcontext (size, placeholder=None, scope=None)
```

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

### classmethod parsexml (node, doc)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

```
phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()
```

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

### resolve()

## resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.

- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

# settext(text)

Sets the text content of the markup element.

```
Parameters text (str) -
```

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
    Alias for text() with strict=True

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=None, normalize_spaces=False)
    Get the text associated with this element (of the specified class)
```

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- retaintokenisation (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

## Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### words (index=None)

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

# Return type str

```
__iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
    ...
```

### \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.List

```
class pynlpl.formats.folia.List(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Element for enumeration/itemisation. Structure element. Contains ListItem elements.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	initianize son.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addidsuffix(idsuffix[, recursive])	added to the parent.  Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	1
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
	Continued on next page

Table 23 – continu	ed from previous page
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)

4.1. Reading FoLiA

Continued on next page

Table 23 – continued from previous page

textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 4
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'List'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n'
XLINK = False
XMLTAG = 'list'
```

## **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

```
ancestors(Class=None)
```

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

# See also:

```
AbstractElement.select()
```

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

### correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

## count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

## hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

## See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.ListItem

```
class pynlpl.formats.folia.ListItem(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Single element in a List. Structure element. Contained within List element.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	tation exists, and it so, now many.

4.1. Reading FoLiA

hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
maphon([ois, street, correctionnanding])	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
items([founditems])	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
*	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
-	

Continued on next page

associated with the element.

Table 24 – continued from previous page

10.000 = 1 0011	area mem previous page
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'List Item'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' n'
XLINK = False
XMLTAG = 'item'
```

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

## annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

# See also:

AbstractElement.select()

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

## count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

## hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

- node XML Element (\*) -
- doc Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Note

```
class pynlpl.formats.folia.Note(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Element used for notes, such as footnotes or warnings or notice blocks.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	tation exists, and it so, now many.

T-1-1- 0F		f		
Table 25 –	CONTINUED	trom	nravinie	nana
Table 25 -	COLITITION	11 0111	DIEVIOUS	pauc

	ued from previous page
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
<pre>incorrection()</pre>	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
<pre>settext(text[, cls]) speech_speaker()</pre>	Retrieves the speaker of the audio or video file asso-
	Retrieves the speaker of the audio or video file associated with the element.
	Retrieves the speaker of the audio or video file asso-

Table 25 – continued from previous page

stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 25
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Note'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = '\n\n'
XLINK = False
XMLTAG = 'note'
```

# **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

```
ancestors(Class=None)
```

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

## Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
     ..
```

# See also:

```
AbstractElement.select()
```

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

# **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

## See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Paragraph

```
class pynlpl.formats.folia.Paragraph(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractStructureElement
```

Paragraph element. A structure element. Represents a paragraph and holds all its sentences (and possibly other structure Whitespace and Quotes).

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	Continued on next page

4.1. Reading FoLiA

Table 26 – continued from previous page

Table 26 – continue	ed from previous page
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
([	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
J (L ) / E J/	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> un-
2 (1 ) 1/	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
X / 1 /	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
4.5	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
	Continued on next page

Continued on next page

Table 26 – continued from previous page

speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xml([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 3
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Paragraph'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n'
```

```
XLINK = False
XMLTAG = 'p'
```

# **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- Class (class) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

# annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

### annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

# Raises

• AllowTokenAnnotation.annotations()

• NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

## count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

## hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

# **Parameters**

- node XML Element (\*) -
- doc Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

# **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Part

```
class pynlpl.formats.folia.Part (doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Generic structure element used to mark a part inside another block.

Do not use this for morphology, use Morpheme instead.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	Continued on next page

Table 27 – continued from previous page

	ed from previous page
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
. 1 / 4/	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
[]	cursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
para annu (noue, uce, noue,	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
phon([els, previousdemmer, strict,])	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
phoneone ([els, concentemanis])	this element (of the specified class).
postappend()	This method will be called after an element is added
postappena()	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
previous([emass, seepe])	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
([meradeemidren, extraaction,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace(clind, args, kwargs)	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
sencences([mack])	cursively) under this element
setdoc(newdoc)	Set a different document.
	Associate a document with this element.
setdocument(doc)	
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
	Continued on next page

Continued on next page

Table 27 – continued from previous page

speech src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
text([cis, retaintokenisation,])	ified class)
(5.1	,
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 35
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Part'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
```

```
XLINK = False
XMLTAG = 'part'
```

# **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

```
alternatives (Class=None, set=None)
```

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

# Raises

• AllowTokenAnnotation.annotations()

• NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

## hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by <code>copy()</code>

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.Quote

Quote: a structure element. For quotes/citations. May hold Word, Sentence or Paragraph data.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
	Continued on next page

hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
masphon([eis, suret, correctionmanding])	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
1/	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
•	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
7 (1.1.1.111	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
(.1.1.1)	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type and set.
resolveword(id)	and set.
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
seriect(Class[, set, recursive, ignore, node]) sentences([index])	Returns a generator of Sentence elements found (re-
Serresires ([mack])	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
occhar cureo()	scop.
	-
settext(text[.cls])	Set the text for this element.
settext(text[, cls]) speech speaker()	Set the text for this element.  Retrieves the speaker of the audio or video file asso-
settext(text[, cls]) speech_speaker()	Retrieves the speaker of the audio or video file asso-

Continued on next page

Table 28 – continued from previous page

	aca nom promoso page
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Quote'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = '\n\n'
XLINK = False
XMLTAG = 'quote'
```

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

## annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

# annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

# See also:

```
AbstractElement.select()
```

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

## hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### Parameters

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Reference

```
class pynlpl.formats.folia.Reference(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractStructureElement
```

A structural element that denotes a reference, internal or external. Examples are references to footnotes, bibliographies, hyperlinks.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	Continued on next page

Table 29 – continued from previous page

	ed from previous page
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
<u>-</u>	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
1 (1)	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
1 3/	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
, , ,	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
21 2 2 2 (1 2) 11 82) 11 11 12 12	places any existing child element of the same type
	and set.
resolve()	
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
Serreences([mack])	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
sechatenes()	_
gottowt(toyt[ als])	Scop.  Set the text for this element
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file associated with the element.
	Continued on next page

Table 29 – continued from previous page

14510 20 001111140	a nom providuo pago
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Reference'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
```

```
XLINK = False
XMLTAG = 'ref'
```

## **Method Details**

## classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

## annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

## Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

### annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

#### See also:

```
AbstractElement.select()
```

# Raises

• AllowTokenAnnotation.annotations()

• NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

## correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

## hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

## resolve()

 $\verb"resolveword"\,(id)$ 

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

#### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

## updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.Row

```
class pynlpl.formats.folia.Row(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractStructureElement
```

# A row in a Table

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
<pre>addtoindex([norecurse])</pre>	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.

4.1. Reading FoLiA

Table 30 – continu	ued from previous page
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
(1.11)	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	Determs the right contest for an element of a list
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
a a t dia a(navidaa)	cursively) under this element Set a different document.
setdoc(newdoc) setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
cottoxt(text[ cls])	scop.  Set the text for this element.
settext(text[, cls])	Retrieves the speaker of the audio or video file asso-
speech_speaker()	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
sheecu_src()	associated with the element.
	Continued on next page

Table 30 – continued from previous page

(F. 4. 7)	
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()
·	·

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Table Row'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n'
XLINK = False
XMLTAG = 'row'
```

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

# See also:

AbstractElement.select()

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

# **Parameters**

- node XML Element (\*) -
- doc Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by <code>copy()</code>

```
settext (text, cls='current')
```

Set the text for this element.

# **Parameters**

- text(str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

```
textvalidation (warnonly=None)
```

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
len ()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

### pynlpl.formats.folia.Sentence

```
class pynlpl.formats.folia.Sentence(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Sentence element. A structure element. Represents a sentence and holds all its words (Word), and possibly other structure such as LineBreak, Whitespace and Quote

# **Method Summary**

init(doc, *args, **kwargs)	Example.
accepts(Class[, raiseexceptions, parentinstance])	1
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
<b>1</b>	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
corrections()	Are there corrections in this sentence?
correctwords(originalwords, newwords,	Generic correction method for words.
**kwargs)	
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
deleteword(word, **kwargs)	TODO: Write documentation
description()	Obtain the description associated with the element.
division()	Obtain the division this sentence is a part of (None
	otherwise).
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	

Table 31 – continued from previous page

	d from previous page
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>insertword(newword, prevword, **kwargs)</pre>	Inserts a word <b>as a correction</b> after an existing word.
<pre>insertwordleft(newword, nextword, **kwargs)</pre>	Inserts a word <b>as a correction</b> before an existing word.
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> under this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
<pre>mergewords(newword, *originalwords, **kwargs)</pre>	TODO: Write documentation
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraph()	Obtain the paragraph this sentence is a part of (None otherwise).
paragraphs([index])	Returns a generator of Paragraph elements found (recursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
	Continued on next page

Continued on next page

Table 31 – continued from previous page

	d from previous page		
replace(child, *args, **kwargs)	Appends a child element like append(), but re-		
	places any existing child element of the same type		
	and set.		
resolveword(id)			
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.		
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.		
sentences([index])	Returns a generator of Sentence elements found (re-		
	cursively) under this element		
setdoc(newdoc)	Set a different document.		
setdocument(doc)	Associate a document with this element.		
setparents()	Correct all parent relations for elements within the		
	scop.		
settext(text[, cls])	Set the text for this element.		
speech_speaker()	Retrieves the speaker of the audio or video file asso-		
	ciated with the element.		
speech_src()	Retrieves the URL/filename of the audio or video file		
	associated with the element.		
<pre>splitword(originalword, *newwords, **kwargs)</pre>	TODO: Write documentation		
stricttext([cls])	Alias for text () with strict=True		
t ext([cls, retaintokenisation,])	Get the text associated with this element (of the spec-		
	ified class)		
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this		
	element (of the specified class).		
textvalidation([warnonly])	Run text validation on this element.		
toktext([cls])	Alias for $text()$ with		
	retaintokenisation=True		
updatetext()	Recompute textual value based on the text content of		
	the children.		
words([index])	the children.  Returns a generator of Word elements found (recur-		
	the children.  Returns a generator of Word elements found (recursively) under this element.		
words([index])  xm1([attribs, elements, skipchildren])	the children.  Returns a generator of Word elements found (recursively) under this element.  Serialises the FoLiA element and all its contents to		
xm1([attribs, elements, skipchildren])	the children.  Returns a generator of Word elements found (recursively) under this element.  Serialises the FoLiA element and all its contents to XML.		
	the children.  Returns a generator of Word elements found (recursively) under this element.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to		
xm1([attribs, elements, skipchildren])	the children.  Returns a generator of Word elements found (recursively) under this element.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to XML.		
<pre>xml([attribs, elements, skipchildren])  xmlstring([pretty_print]) iter()</pre>	the children.  Returns a generator of Word elements found (recursively) under this element.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to XML.  Iterate over all children of this element.		
<pre>xml([attribs, elements, skipchildren]) xmlstring([pretty_print])</pre>	the children.  Returns a generator of Word elements found (recursively) under this element.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to XML.  Iterate over all children of this element.  Returns the number of child elements under the cur-		
<pre>xml([attribs, elements, skipchildren])  xmlstring([pretty_print]) iter()</pre>	the children.  Returns a generator of Word elements found (recursively) under this element.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to XML.  Iterate over all children of this element.		

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlpl.ANNOTATIONTYPE = 8

AUTH = True

AUTO_GENERATE_ID = True
```

LABEL = 'Sentence'

OCCURRENCES = 0

OCCURRENCES\_PER\_SET = 0

```
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' '
XLINK = False
XMLTAG = 's'
Method Details
___init___(doc, *args, **kwargs)
    Example:
    sentence = paragraph.append( folia.Sentence)
    sentence.append( folia.Word, 'This')
    sentence.append(folia.Word, 'is')
    sentence.append( folia.Word, 'a')
    sentence.append( folia.Word, 'test', space=False)
    sentence.append( folia.Word, '.')
    Example:
    sentence = folia.Sentence( doc, folia.Word(doc, 'This'), folia.Word(doc, 'is
    →'), folia.Word(doc, 'a'), folia.Word(doc, 'test', space=False), folia.
    →Word(doc, '.') )
    paragraph.append(sentence)
    See also:
    AbstractElement.__init__()
__init__ (doc, *args, **kwargs)
    Example:
    sentence = paragraph.append( folia.Sentence)
    sentence.append( folia.Word, 'This')
    sentence.append(folia.Word, 'is')
    sentence.append( folia.Word, 'a')
    sentence.append( folia.Word, 'test', space=False)
    sentence.append( folia.Word, '.')
```

Example:

```
sentence = folia.Sentence( doc, folia.Word(doc, 'This'), folia.Word(doc, 'is
    →'), folia.Word(doc, 'a'), folia.Word(doc, 'test', space=False), folia.
    →Word(doc, '.') )
paragraph.append(sentence)
```

#### See also:

```
AbstractElement.__init__()
```

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

## annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

# See also:

```
AbstractElement.select()
```

#### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
     See AbstractElement.append()
context (size, placeholder=None, scope=None)
     Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy (newdoc=None, idsuffix=")
     Make a deep copy of this element and all its children.
         Parameters
             • newdoc (Document) – The document the copy should be associated with.
             • idsuffix (str or bool) - If set to a string, the ID of the copy will be append with
               this (prevents duplicate IDs when making copies for the same document). If set to True,
               a random suffix will be generated.
         Returns a copy of the element
copychildren (newdoc=None, idsuffix=")
     Generator creating a deep copy of the children of this element.
     Invokes copy () on all children, parameters are the same.
correct (**kwargs)
     Apply a correction (TODO: documentation to be written still)
corrections()
     Are there corrections in this sentence?
         Returns bool
correctwords (originalwords, newwords, **kwargs)
     Generic correction method for words. You most likely want to use the helper functions Sentence.
     splitword(), Sentence.mergewords(), deleteword(), insertword() instead
count (Class, set=None, recursive=True, ignore=True, node=None)
     Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
         Returns int
deepvalidation()
     Perform deep validation of this element.
         Raises DeepValidationError
deleteword(word, **kwargs)
     TODO: Write documentation
description()
     Obtain the description associated with the element.
         Raises NoSuchAnnotation if there is no associated description.
division()
     Obtain the division this sentence is a part of (None otherwise). Shortcut for AbstractElement.
     ancestor()
feat (subset)
```

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### generate\_id(cls)

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

#### hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

# $\verb|hastext| (cls='current', strict=True, correction handling=1)|$

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
insertword (newword, prevword, **kwargs)
```

Inserts a word **as a correction** after an existing word.

This method automatically computes the index of insertion and calls AbstractElement.insert()

#### **Parameters**

- **newword** (Word) The new word to insert
- **prevword** (*Word*) The word to insert after

**Keyword Arguments suggest** (bool) – Do a suggestion for correction rather than the default authoritive correction

# See also:

 $AbstractElement.insert () \ and \ AbstractElement.getindex () \ If \ you \ do \ not \ want \ to \ do \ corrections$ 

# insertwordleft (newword, nextword, \*\*kwargs)

Inserts a word **as a correction** before an existing word.

```
Reverse of Sentence.insertword().
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

# Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
mergewords (newword, *originalwords, **kwargs)
```

TODO: Write documentation

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

#### paragraph()

Obtain the paragraph this sentence is a part of (None otherwise). Shortcut for AbstractElement. ancestor()

### paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.

• correctionhandling — Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon - if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

# previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

• Class (\*) - The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all

• **scope** (\*) – A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None) Select child elements of the specified class.
```

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

4.1. Reading FoLiA

```
sentences (index=None)
```

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
splitword(originalword, *newwords, **kwargs)
```

TODO: Write documentation

```
stricttext (cls='current')
```

```
Alias for text () with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

## words (index=None)

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

#### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
    ...
```

```
___len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Table

A table consisting of Row elements that in turn consist of Cell elements

# **Method Summary**

(1	T., '4', 1', 1C
init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
	Continued on next page

Table 32 – continued from previous page

	ed from previous page
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
anossos ( <b>Sia</b> ssos)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
ances cors([Class])	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
annocacions(Ciassi, setj)	class.
1/-1:14 * **1	
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
5(1.1)	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
get index(clind[, recursive, ignore])	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
getmetadata([Key])	
water at 1-1 to the offerteint lander with a 1	matically inherited from parent elements
gettextdelimiter([retaintokenisation])	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
7 7 7 7 7 7 7 7	tation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
<pre>hasphon([cls, strict, correctionhandling])</pre>	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
<del></del>	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
	Continued on next page
	Continued on next page

4.1. Reading FoLiA

T 11 00		•		
Table 32 –	CONTINUED	trom	nravinie	nana
Table oz	CONTINUCA	11 0111	picvious	page

	ued from previous page
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
1 / 1/	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
[ ([ ])	cursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
[1.01.0.0.000, 0.000, 0.000]	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
phon([elo, previousaemmer, suret,])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
phoneometre ([cis, correctionnanding])	this element (of the specified class).
nost annond()	This method will be called after an element is added
postappend()	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
prevrous([Class, scope])	type and if it does not cross the boundary of the de-
	* ·
relaxng([includechildren, extraattribs,])	fined scope.  Returns a RelaxNG definition for this element (as an
retaxng([includecinidren, extraatinos,])	
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
4.0	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
(5.7)	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
(F1)	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
	Continued on next page
	OUTILITIES OF TICKLE DAGE

Continued on next page

347

Table 32 – continued from previous page

	1 1 3
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 33
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Table'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n \n'
XLINK = False
XMLTAG = 'table'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

4.1. Reading FoLiA

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

# classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by COPY ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation(type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

```
AllowTokenAnnotation.annotations() AbstractElement.select()
```

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

### See also:

```
AbstractElement.select()
```

# **Raises**

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

# **Parameters**

• newdoc (Document) – The document the copy should be associated with.

• idsuffix (str or bool) – If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

# ${\tt generate\_id}\,(\mathit{cls})$

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### **Returns** int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

**hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

items (founditems=[])

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

json (attribs=None, recurse=True, ignorelist=False)

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

```
leftcontext (size, placeholder=None, scope=None)
```

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
paragraphs (index=None)
```

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

# **Parameters**

- **cls** (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.

- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

```
for sense in text.select(folia.Sense, 'cornetto', True, [folia.Original,

→folia.Suggestion, folia.Alternative] ):
...
```

### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

# speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### words (index=None)

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
```

## \_\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Term

```
class pynlpl.formats.folia.Term(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

A term, often used in contect of Entry

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
	Continued on next page

T !! 00		•		
Table 33 -	CONTINUED	trom	nravinie	nage
Table 00	CONTINUCA	11 0111	picvious	page

	d from previous page
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
6.7	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
guere, nearly	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
11111111111111111111111111111111111111	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
geemaen(emiat, recarsive, ignore))	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
geemeeadaea([Rey])	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
nasamioeaeron(Olusse, seel)	tation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
masphon([cis, suret, confectionnaliding])	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
THEOTTECTION	
	the Correction element (evaluating to True), otherwise it returns None
in a ant (index shild *area **lowers)	WISE IL IELUIIIS INOIIE
insert(index, child, *args, **kwargs)	Dotume a double first flot list of all towns halos (1.)
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
	Continued on next page

Table 33 – continued from previous page

	led from previous page		
layers([annotationtype, set])	Returns a list of annotation layers found directly un		
	der this element, does not include alternative layers		
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.		
next([Class, scope, reverse])	Returns the next element, if it is of the specified type		
	and if it does not cross the boundary of the defined		
	scope.		
originaltext([cls])	Alias for retrieving the original uncorrect text.		
paragraphs([index])	Returns a generator of Paragraph elements found (re-		
	cursively) under this element.		
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-		
	ment into an instance of the Class.		
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this		
	element (of the specified class)		
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with		
<u></u>	this element (of the specified class).		
postappend()	This method will be called after an element is added		
r	to another and does some checks.		
previous([Class, scope])	Returns the previous element, if it is of the specified		
previous (remass, seeper)	type and if it does not cross the boundary of the de-		
	fined scope.		
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an		
reraxing (includeeimaren, extrautilios,)	XML element (lxml.etree) rather than a string)		
remove(child)	Removes the child element		
replace(child, *args, **kwargs)			
reprace(clind, digs, kwaigs)	Appends a child element like append(), but replaces any existing child element of the same type		
	and set.		
resolveword(id)	and set.		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.		
right context(size[, placeholder, scope])			
and a st (Class set requesive ignore node)			
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.		
<pre>select(Class[, set, recursive, ignore, node]) sentences([index])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (re-		
sentences([index])	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element		
sentences([index]) setdoc(newdoc)	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.		
sentences([index]) setdoc(newdoc) setdocument(doc)	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.		
sentences([index]) setdoc(newdoc)	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the		
sentences([index])  setdoc(newdoc) setdocument(doc) setparents()	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.		
sentences([index])  setdoc(newdoc) setdocument(doc) setparents()	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file asso-		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the spec-		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the spec-		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_speaker()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True		
<pre>sentences([index])  setdoc(newdoc) setdocument(doc) setparents()  settext(text[, cls]) speech_speaker()  speech_speaker()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	Select child elements of the specified class.  Returns a generator of Sentence elements found (recursively) under this element  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with		

Continued on next page

Table 33 – continued from previous page

words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 38
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Term'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n \n'
XLINK = False
XMLTAG = 'term'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by COPY ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- set (str) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation(type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

## Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

```
AllowTokenAnnotation.annotations() AbstractElement.select()
```

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

### See also:

```
AbstractElement.select()
```

## **Raises**

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

## **Parameters**

• **newdoc** (*Document*) – The document the copy should be associated with.

• idsuffix (str or bool) – If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

### copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

### findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

# ${\tt generate\_id}\,(\mathit{cls})$

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### **Returns** int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasannotation (Class, set=None)
```

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

## hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

```
leftcontext (size, placeholder=None, scope=None)
```

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.

- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by COPY ()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### words (index=None)

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

#### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
__iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
    ...
```

```
___len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.TableHead

```
class pynlpl.formats.folia.TableHead(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractStructureElement
```

Encapsulated the header of a table, contains Cell elements

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
	Continued on next page

Table 34 – continued from previous page

	ed from previous page
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
annocaciono(Classi, seci)	class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
• •	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	*
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
J = 1 ( 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
g 0 0 m 0 0 a a a 0 a ([110]])	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
masammoeaerom(Classe, seej)	tation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
	Continued on next page

4.1. Reading FoLiA

T 11 04		•		
Table 34 –	CONTINUED	trom	nrevinis	nage
IUDIC OT	COLITICICA	11 0111	providuo	page

	led from previous page	
layers([annotationtype, set])	Returns a list of annotation layers found directly un-	
	der this element, does not include alternative layer	
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.	
next([Class, scope, reverse])	Returns the next element, if it is of the specified type	
	and if it does not cross the boundary of the defined	
	scope.	
originaltext([cls])	Alias for retrieving the original uncorrect text.	
paragraphs([index])	Returns a generator of Paragraph elements found (re-	
	cursively) under this element.	
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-	
	ment into an instance of the Class.	
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this	
	element (of the specified class)	
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with	
_	this element (of the specified class).	
postappend()	This method will be called after an element is added	
<del>-</del>	to another and does some checks.	
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified	
- · · · · · · · · · · · · · · · · · · ·	type and if it does not cross the boundary of the de-	
	fined scope.	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
sentences([index])	Returns a generator of Sentence elements found (re-	
	cursively) under this element	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for $text()$ with	
\L	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
	Continued on next page	

Continued on next page

Table 34 – continued from previous page

	9 -
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Table Header'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' \n \n'
XLINK = False
XMLTAG = 'tablehead'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

4.1. Reading FoLiA 373

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by COPY()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- set (str) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation(type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

## Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

```
AllowTokenAnnotation.annotations() AbstractElement.select()
```

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ...
```

### See also:

```
AbstractElement.select()
```

# Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
See AbstractElement.append()
```

```
context(size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

## **Parameters**

• **newdoc** (*Document*) – The document the copy should be associated with.

• idsuffix (str or bool) – If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

### copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

# findcorrection handling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

# ${\tt generate\_id}\,(\mathit{cls})$

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### **Returns** int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

**hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

items (founditems=[])

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

json (attribs=None, recurse=True, ignorelist=False)

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

```
leftcontext (size, placeholder=None, scope=None)
```

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

# **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.

- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

```
for sense in text.select(folia.Sense, 'cornetto', True, [folia.Original,

→folia.Suggestion, folia.Alternative] ):
...
```

### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

# speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### words (index=None)

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

#### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
```

## \_\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Text

```
class pynlpl.formats.folia.Text(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

A full text. This is a high-level element (not to be confused with TextContent!). This element may contain <code>Division</code>,:class:\*Paragraph, class:\*Sentence, etc..

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
•	0

Continued on next page

T		•		
Table 35 –	CONTINUED	trom	nravinie	nana
Table 00	COLITICIACA	11 0111	picvious	page

	ed from previous page
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
, 1)	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
ances cor ( Classes)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
ancestors([Class])	
(1 - 1 - (1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
copy chiliar chi([he wase, rasahmi])	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
correct ( kwaigs)	written still)
Close set manufacine ignore model)	
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
geethaen(emat, recursive, ignore <sub>1</sub> )	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
geemeeadaea([hey])	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
<u> </u>	
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
<pre>hasphon([cls, strict, correctionhandling])</pre>	Does this element have phonetic content (of the spec-
	ified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
items([founditems])	Returns a depth-first flat list of all items below this
1 como ([rounditomo])	element (not limited to AbstractElement)
	Continued on next page
	Ontinued on heat hade

Table 35 – continued from previous page
-----------------------------------------

	ued from previous page
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
==-2 === ([	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
	Returns the next element, if it is of the specified type
next([Class, scope, reverse])	and if it does not cross the boundary of the defined
( 1 1 )	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
	cursively) under this element.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
-	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
T (C) () I	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
prioriconiconic([vis, concentialidinig])	this element (of the specified class).
postappend()	This method will be called after an element is added
postappenaty	to another and does some checks.
([0]	
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sentences([index])	Returns a generator of Sentence elements found (re-
([])	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
* /	Correct all parent relations for elements within the
setparents()	•
(4) (F. 1.2)	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
ceacconcency[cis, conectionnanding])	element (of the specified class).
+	
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
	Continued on next page

Continued on next page

Table 35 – continued from previous page

Table 66 C	ortinaca nom previous page
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Text Body'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = '\n\n'
XLINK = False
XMLTAG = 'text'
Method Details
__init__ (doc, *args, **kwargs)
```

Initialize self. See help(type(self)) for accurate signature.

```
___init___(doc, *args, **kwargs)
```

Initialize self. See help(type(self)) for accurate signature.

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

## Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

```
AllowTokenAnnotation.annotations() AbstractElement.select()
```

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

## Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

# See also:

```
AbstractElement.select()
```

# Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

## correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

#### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

# Parameters

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

#### json (attribs=None, recurse=True, ignorelist=False)

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext(cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

# $\verb"resolveword"\,(id)$

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
```

```
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *text* (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

Returns The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

## updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

#### words (index=None)

Returns a generator of Word elements found (recursively) under this element.

**Parameters index** (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
iter ()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

### \_\_str\_\_()

Alias for text ()

# pynlpl.formats.folia.Whitespace

```
class pynlpl.formats.folia.Whitespace(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractStructureElement
```

Whitespace element, signals a vertical whitespace

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	

Table 36 -	continued	from	previous page	
iabic co	COLITICICA	11 0111	providus page	

	ed from previous page
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally to all child IDs as well.
<pre>addtoindex([norecurse])</pre>	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	

Table 36 – continued from previous page	Table 36 –	continued	from	previous	page
-----------------------------------------	------------	-----------	------	----------	------

	ued from previous page
items([founditems])	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found directly un-
- · · · · · · · · · · · · · · · · · · ·	der this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
1	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraphs([index])	Returns a generator of Paragraph elements found (re-
T. 1. 2. 2. 2. 1. ([	cursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
parsonni (nout, uot, ninago)	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
phon([els, previousaemmer, strict,])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
phoneometric([cis, correctionnanding])	this element (of the specified class).
postappend()	This method will be called after an element is added
postappena()	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
previous([Class, scope])	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
reraxing([includecimaten, extraatinos,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	
reprace(ciliu, raigs, rewaigs)	Appends a child element like append(), but re-
	places any existing child element of the same type and set.
resolveword(id)	and set.
rightcontext(size[, placeholder, scope])	Detume the might context for an element, as a list
select(Class[, set, recursive, ignore, node])	Returns the right context for an element, as a list.  Select child elements of the specified class.
<u> </u>	
sentences([index])	Returns a generator of Sentence elements found (re-
( 1 ( 1 )	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
( 11)	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
·	Continued on next page

Table 36 – continued from previous page

Table 00 Continue	ca nom previous page
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 6
AUTH = True
AUTO_GENERATE_ID = True
LABEL = 'Whitespace'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ''
XLINK = False
XMLTAG = 'whitespace'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

4.1. Reading FoLiA 399

```
__init__ (doc, *args, **kwargs)
```

Initialize self. See help(type(self)) for accurate signature.

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

## classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

# Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

```
AllowTokenAnnotation.annotations() AbstractElement.select()
```

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

# See also:

```
AbstractElement.select()
```

### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str* or *bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

### sentences (index=None)

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
```

```
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=None, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *text* (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### words (index=None)

Returns a generator of Word elements found (recursively) under this element.

**Parameters index** (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

### \_\_str\_\_()

Alias for text ()

# pynlpl.formats.folia.Word

```
class pynlpl.formats.folia.Word(doc, *args, **kwargs)
```

Bases: pynlpl.formats.folia.AbstractStructureElement, pynlpl.formats.folia. AllowCorrections

Word (aka token) element. Holds a word/token and all its related token annotations.

# **Method Summary**

init(doc, *args, **kwargs)	Constructor for words.		
accepts(Class[, raiseexceptions, parentinstance])			

Table 37 – continued from previous page

	ued from previous page
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
, J	specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
anococoro((crass <sub>1</sub> )	fectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain a single annotation element.  Obtain child elements (annotations) of the specified
	class.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
division()	Obtain the deepest division this word is a part of, otherwise return None
domain([set])	Shortcut: returns the FoLiA class of the domain an-
aomarn([set])	notation (will return only one if there are multiple!)
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspans(type[, set])	Yields span annotation elements of the specified type that include this word.
generate_id(cls)	and morace only north
getcorrection([set, cls])	
getcorrections([set, cls])	
<u> </u>	Get the index at which an element occurs, recursive
<pre>getindex(child[, recursive, ignore])</pre>	by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Returns the text delimiter
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
	Continued on next page

Table 37 – continued from previous page

	ed from previous page
hasannotationlayer([annotationtype, set])	Does the specified annotation layer exist?
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
layers([annotationtype, set])	Returns a list of annotation layers found <i>directly</i> under this element, does not include alternative layers
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
lemma([set])	Shortcut: returns the FoLiA class of the lemma annotation (will return only one if there are multiple!)
morpheme(index[, set])	Returns a specific morpheme, the n'th morpheme (given the particular set if specified).
morphemes([set])	Generator yielding all morphemes (in a particular set if specified).
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
paragraph()	Obtain the paragraph this word is a part of, otherwise return None
paragraphs([index])	Returns a generator of Paragraph elements found (recursively) under this element.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML element into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with this element (of the specified class).
phoneme(index[, set])	Returns a specific phoneme, the n'th morpheme (given the particular set if specified).
phonemes([set])	Generator yielding all phonemes (in a particular set if specified).
pos([set])	Shortcut: returns the FoLiA class of the PoS annotation (will return only one if there are multiple!)
postappend()	This method will be called after an element is added to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
	Continued on next page

Table 37 – continued from previous page
-----------------------------------------

	ed from previous page
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
sense([set])	Shortcut: returns the FoLiA class of the sense anno-
	tation (will return only one if there are multiple!)
sentence()	Obtain the sentence this word is a part of, otherwise
	return None
sentences([index])	Returns a generator of Sentence elements found (re-
	cursively) under this element
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
- *	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
split(*newwords, **kwargs)	
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
words([index])	Returns a generator of Word elements found (recur-
	sively) under this element.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
	Returns the number of child elements under the cur-
-	rent element.
str()	Alias for text ()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = 1
AUTH = True
AUTO_GENERATE_ID = True
```

LABEL = 'Word/Token'

```
OCCURRENCES = 0
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = ' '
XLINK = False
XMLTAG = 'w'
Method Details
__init__(doc, *args, **kwargs)
    Constructor for words.
    See AbstractElement.__init__ for all inherited keyword arguments and parameters.
    Keyword arguments:
      • space (bool): Indicates whether this token is followed by a space (defaults to True)
    Example:
    sentence.append( folia.Word, 'This')
    sentence.append( folia.Word, 'is')
    sentence.append( folia.Word, 'a')
    sentence.append( folia.Word, 'test', space=False)
    sentence.append( folia.Word, '.')
    See also:
    AbstractElement.___init__
 __init___(doc, *args, **kwargs)
    Constructor for words.
    See AbstractElement.__init__ for all inherited keyword arguments and parameters.
    Keyword arguments:
      • space (bool): Indicates whether this token is followed by a space (defaults to True)
    Example:
```

```
sentence.append( folia.Word, 'This')
sentence.append( folia.Word, 'is')
sentence.append( folia.Word, 'a')
sentence.append( folia.Word, 'test', space=False)
sentence.append( folia.Word, '.')
```

# See also:

```
AbstractElement.___init__
```

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

## classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

## **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (*str*) The set you want to retrieve (defaults to None, which selects irregardless of set)

Yields Alternative elements

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Obtain a single annotation element.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

## Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

# annotations (Class, set=None)

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

# Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

# See also:

```
AbstractElement.select()
```

### Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## division()

Obtain the deepest division this word is a part of, otherwise return None

# domain (set=None)

Shortcut: returns the FoLiA class of the domain annotation (will return only one if there are multiple!)

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### **findspans** (*type*, *set=None*)

Yields span annotation elements of the specified type that include this word.

#### **Parameters**

- type The annotation type, can be passed as using any of the AnnotationType member, or by passing the relevant AbstractSpanAnnotation or AbstractAnnotationLayer class.
- set (str or None) Constrain by set

# Example:

Yields Matching span annotation instances (derived from AbstractSpanAnnotation)

```
generate_id (cls)
getcorrection (set=None, cls=None)
getcorrections (set=None, cls=None)
getindex (child, recursive=True, ignore=True)
    Get the index at which an element occurs, recursive by default!
```

Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

```
gettextdelimiter (retaintokenisation=False)
```

Returns the text delimiter

# hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# hasannotationlayer (annotationtype=None, set=None)

Does the specified annotation layer exist?

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling — Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

## Returns dict

# layers (annotationtype=None, set=None)

Returns a list of annotation layers found directly under this element, does not include alternative layers

```
leftcontext (size, placeholder=None, scope=None)
```

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
lemma (set=None)
```

Shortcut: returns the FoLiA class of the lemma annotation (will return only one if there are multiple!)

### morpheme (index, set=None)

Returns a specific morpheme, the n'th morpheme (given the particular set if specified).

### morphemes (set=None)

Generator yielding all morphemes (in a particular set if specified). For retrieving one specific morpheme by index, use morpheme() instead

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# paragraph()

Obtain the paragraph this word is a part of, otherwise return None

## paragraphs (index=None)

Returns a generator of Paragraph elements found (recursively) under this element.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the generator of all

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.

• correctionhandling - Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

## phoneme (index, set=None)

Returns a specific phoneme, the n'th morpheme (given the particular set if specified).

## phonemes (set=None)

Generator yielding all phonemes (in a particular set if specified). For retrieving one specific morpheme by index, use morpheme() instead

# pos (set=None)

Shortcut: returns the FoLiA class of the PoS annotation (will return only one if there are multiple!)

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

# ${\tt resolveword}\,(id)$

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

### sense (set=None)

Shortcut: returns the FoLiA class of the sense annotation (will return only one if there are multiple!)

### sentence()

Obtain the sentence this word is a part of, otherwise return None

```
sentences (index=None)
```

Returns a generator of Sentence elements found (recursively) under this element

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning a generator of all

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
split (*newwords, **kwargs)
stricttext (cls='current')
    Alias for text () with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- ${\tt cls}\,({\it str})$  The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

## See also:

text() phoncontent() phon()

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
words (index=None)
```

Returns a generator of Word elements found (recursively) under this element.

**Parameters** index (\*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

AbstractElement.xmlstring() - for direct string output

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
__iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

The FoLiA documentation explains the exact semantics and use of these in detail. Make sure to consult it to familiarize yourself with how the elements should be used.

FoLiA and this library enforce explicit rules about what elements are allowed in what others. Exceptions will be raised when this is about to be violated.

# 4.1.10 Common attributes

The FoLiA paradigm features *sets* and *classes* as primary means to represent the actual value (class) of an annotation. A set often corresponds to a tagset, such as a set of part-of-speech tags, and a class is one selected value in such a set.

The paradigm furthermore introduces other common attributes to set on annotation elements, such as an identifier, information on the annotator, and more. A full list is provided below:

- element . id (str) The unique identifier of the element
- element.set (str) The set the element pertains to.
- element.cls (str) The assigned class, i.e. the actual value of the annotation, defined in the set. Classes correspond with tagsets in this case of many annotation types. Note that since *class* is already a reserved keyword in python, the library consistently uses cls everywhere.
- element .annotator (str) The name or ID of the annotator who added/modified this element
- element.annotatortype-The type of annotator, can be either folia.AnnotatorType.MANUAL or folia.AnnotatorType.AUTO
- element.confidence (float) A confidence value expressing
- element . datetime (datetime.datetime) The date and time when the element was added/modified.
- element.n (str) An ordinal label, used for instance in enumerated list contexts, numbered sections, etc..

The following attributes are specific to a speech context:

- element.src(str) A URL or filename referring the an audio or video file containing the speech. Access this attribute using the element.speaker\_src() method, as it is inheritable from ancestors.
- element.speaker (str) The name of ID of the speaker. Access this attribute using the element. speech\_speaker() method, as it is inheritable from ancestors.
- element.begintime (4-tuple) The time in the above source fragment when the phonetic content of this element starts, this is a (hours, minutes, seconds, milliseconds) tuple.
- element.endtime (4-tuple) The time in the above source fragment when the phonetic content of this element ends, this is a (hours, minutes, seconds, milliseconds) tuple.

Attributes that are not available for certain elements, or not set, default to None.

# 4.1.11 Annotations

As FoLiA is a format for linguistic annotation, accessing annotation is one of the primary functions of this library. This can be done using the methods <code>AllowTokenAnnotation.annotations()</code> or <code>AllowTokenAnnotation.annotation()</code> that are available on many FoLiA elements. These methods are similar to the <code>AbstractElement.select()</code> method except they will raise a <code>NoSuchAnnotation()</code> is that the former will grab only one and raise an exception if there are more between which it can't disambiguate, whereas the second is a generator, but will still raise an exception if none is found:

```
for word in doc.words():
    try:
        pos = word.annotation(folia.PosAnnotation, 'http://somewhere/CGN')
        lemma = word.annotation(folia.LemmaAnnotation)
        print("Word: ", word)
        print("ID: ", word.id)
        print("Pos-tag: ", pos.cls)
        print("Pos Annotator: ", pos.annotator)
```

(continues on next page)

(continued from previous page)

```
print("Lemma-tag: " , lemma.cls)
except folia.NoSuchAnnotation:
    print("No PoS or Lemma annotation")
```

Note that the second argument of <code>AllowTokenAnnotation.annotation()</code>, <code>AllowTokenAnnotation.annotation()</code>, <code>AllowTokenAnnotation.annotation()</code> or <code>AbstractElement.select()</code> can be used to restrict your selection to a certain set. In the above example we restrict ourselves to Part-of-Speech tags in the CGN set.

# **Token Annotation Types**

The following token annotation elements are available in FoLiA, they are embedded under a structural element (not necessarily a token, despite the name).

DomainAnnotation	Domain annotation: an extended token annotation ele-
	ment
PosAnnotation	Part-of-Speech annotation: a token annotation element
LangAnnotation	Language annotation: an extended token annotation el-
	ement
LemmaAnnotation	Lemma annotation: a token annotation element
SenseAnnotation	Sense annotation: a token annotation element
SubjectivityAnnotation	Subjectivity annotation/Sentiment analysis: a token an-
	notation element

# pynlpl.formats.folia.DomainAnnotation

class pynlpl.formats.folia.DomainAnnotation(doc, \*args, \*\*kwargs)
 Bases: pynlpl.formats.folia.AbstractExtendedTokenAnnotation

Domain annotation: an extended token annotation element

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
	Continued on next page

Table 39 – continued from previous page

	ued from previous page
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
<pre>postappend()</pre>	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
	Continued on next page

Table 39 – continued from previous page

	ued from previous page
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.Tannotationtype = 11

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Domain'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = False
```

```
REQUIRED\_ATTRIBS = (1,)
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'domain'
Method Details
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
             from. Not instances!
```

4.1. Reading FoLiA

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str or bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.PosAnnotation

```
class pynlpl.formats.folia.PosAnnotation(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractTokenAnnotation
```

Part-of-Speech annotation: a token annotation element

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

Continued on next page

Table 40 -	continued	from	previous page
Iable 40 -	CONTINUEU	11 0111	pievious page

Table 40 – continu	ued from previous page
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	1
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
	Continued on next page

Table 40 – continued from previous page

	Determine Dela NC de Caldan Carabia de caracteria	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.	
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xml([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
•	rent element.	
str()	Alias for text()	

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 9

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Part-of-Speech'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 1

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False
```

```
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = (1,)
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'pos'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
```

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select

from. Not instances!

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

# Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

# hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.LangAnnotation

```
class pynlpl.formats.folia.LangAnnotation(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractExtendedTokenAnnotation
```

Language annotation: an extended token annotation element

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

Continued on next page

Table 41 – continued from previous page

	d from previous page
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
, 1, 1,	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	<del>-</del>
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
J. 1. ([	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
(L) (1) (1) (1) (1) (1)	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
· , , ,	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
2 X /1 / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / / /	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
2 (2)	this element (of the specified class).
	` 1 /
postappend()	This method will be called after an element is added
postappend()	This method will be called after an element is added to another and does some checks.
	to another and does some checks.
<pre>postappend() previous([Class, scope])</pre>	to another and does some checks.  Returns the previous element, if it is of the specified
	to another and does some checks.

Table 41 – continued from previous page

	ed from previous page
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 31

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Language'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 1

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False
```

```
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = (1,)
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'lang'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
```

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select

from. Not instances!

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### **Returns** int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
\begin{tabular}{ll} \textbf{text} (cls='current', retaintokenisation=False, previous delimiter=", strict=False, correction handling=1, normalize\_spaces=False) \\ \end{tabular}
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.LemmaAnnotation

```
class pynlpl.formats.folia.LemmaAnnotation(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractTokenAnnotation
```

Lemma annotation: a token annotation element

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

Continued on next page

Table 12 -	continued	from	previous page
Iable 42 -	CONTINUEU	11 0111	previous page

	ued from previous page
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
<pre>getmetadata([key])</pre>	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
**************************************	wise it returns None
insert(index, child, *args, **kwargs)	Determine a death foot flat list of all items halow this
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
descriptions recovered impossibility	Serialises the FoLiA element and all its contents to a
json([attribs, recurse, ignorelist])	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
next([class, scope, reverse])	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexm1(node, doc, **kwargs)	Internal class method used for turning an XML ele-
parseaux (node, doc, kwaigs)	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
priori([ois, proviousuominior, strict,])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
proceeding ([clos, correction and mag])	this element (of the specified class).
postappend()	This method will be called after an element is added
r	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
received ([Cimos, scope])	type and if it does not cross the boundary of the de-
	fined scope.
	Continued on next page
	Commission on none page

Table 42 – continued from previous page

	Determine Dela NG de Caldan Carda a la mante	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.	
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xml([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
•	rent element.	
str()	Alias for text()	
<del></del>		

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 10

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Lemma'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 1

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False
```

```
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = (1,)
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'lemma'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
```

from. Not instances!

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

# Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

# hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

## Return type str

```
__iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.SenseAnnotation

```
class pynlpl.formats.folia.SenseAnnotation(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractTokenAnnotation
```

Sense annotation: a token annotation element

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

Chapter 4. FoLiA library

Table 43 – continued from previous page

Table 43 – continued	d from previous page
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
<pre>hasphon([cls, strict, correctionhandling])</pre>	Does this element have phonetic content (of the spec-
	ified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	Get the phonetic content explicitly associated with this element (of the specified class).
<pre>phoncontent([cls, correctionhandling]) postappend()</pre>	Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added
postappend()	Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.
	Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified
postappend()	Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.

4.1. Reading FoLiA

Table 43 – continued from previous page

	ed from previous page
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 12

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Semantic Sense'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False
```

```
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = (1,)
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'sense'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
```

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select

from. Not instances!

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

### context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

## hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• **cls** (*str*) – The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter___()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.SubjectivityAnnotation

```
class pynlpl.formats.folia.SubjectivityAnnotation(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractTokenAnnotation
```

Subjectivity annotation/Sentiment analysis: a token annotation element

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

Continued on next page

Table 11		f.,	
1able 44 –	continuea	Irom	previous page

Table 44 – continu	led from previous page
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	*
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
	Continued on next page

Table 44 – continued from previous page

	Data was Data NG da Gaiding Conditional account (constant
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xml([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
-	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 19

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Subjectivity/Sentiment'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 1

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False
```

```
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = (1,)
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'subjectivity'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
```

from. Not instances!

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

### context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

## hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

### xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

# \_\_str\_\_()

Alias for text ()

## Text and phonetic annotation

The actual text of an element, or a phonetic textual representation, are also considered annotations themselves.

TextContent	Text content element (t), holds text to be associated with whatever element the text content element is a child of.
PhonContent	Phonetic content element (ph), holds a phonetic representation to be associated with whatever element the phonetic content element is a child of.

### pynlpl.formats.folia.TextContent

```
class pynlpl.formats.folia.TextContent(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Text content element (t), holds text to be associated with whatever element the text content element is a child of

Text content elements on structure elements like *Paragraph* and *Sentence* are by definition untokenised. Only on Word` level and deeper they are by definition tokenised.

Text content elements can specify offset that refer to text at a higher parent level. Use the following keyword

## arguments:

- ref=: The instance to point to, this points to the element holding the text content element, not the text content element itself.
- $\bullet$  offset=: The offset where this text is found, offsets start at 0

# **Method Summary**

init(doc, *args, **kwargs)	Example.
accepts(Class[, raiseexceptions, parentinstance])	Zamipie.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addas 20(pareint, set, raiseenceptions)	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
, , ,	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
finddefaultreference()	Find the default reference for text offsets: The par-
	ent of the current textcontent's parent (counting only
	Structure Elements and Subtoken Annotation Ele-
C' 1 7 7 7 7 4 9 4 1	ments)
findreplaceables(parent, set, **kwargs)	(Method for internal usage, see AbstractElement)
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
geemetadata([ncy])	matically inherited from parent elements
<pre>getreference([validate])</pre>	Returns and validates the Text Content's reference.
<pre>getTeTeTeTeTeTeTeTeTeTeTeTeTeTeTeTeTeTeT</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
macphonicion, suret, correctionnuming])	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
consequence of the control of	Continued on next page
	Continuou on noxt page

4.1. Reading FoLiA

	ued from previous page	
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other-	
	wise it returns None	
<pre>insert(index, child, *args, **kwargs)</pre>	wise it returns none	
items([founditems])	Returns a depth-first flat list of <i>all</i> items below this	
rems([rounditems])	element (not limited to AbstractElement)	
json([attribs, recurse, ignorelist])	See AbstractElement.json()	
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.	
next([Class, scope, reverse])	Returns the next element, if it is of the specified type	
mext([class, scope, levelse])	and if it does not cross the boundary of the defined	
originaltext([cls])	scope.  Alias for retrieving the original uncorrect text.	
parsexml(node, doc, **kwargs)	(Method for internal usage, see AbstractElement)	
phon([cls, previousdelimiter, strict,])		
phon([cis, previousdenimer, strict,])	Get the phonetic representation associated with this	
phoncontent([cls, correctionhandling])	element (of the specified class)  Get the phonetic content explicitly associated with	
phoneomeenc([cis, confectionnandinig])	this element (of the specified class).	
postappend()	This method will be called after an element is added	
postappenu()	to another and does some checks.	
previous([Class, scope])	Returns the previous element, if it is of the specified	
previous([Class, scope])	type and if it does not cross the boundary of the de-	
	fined scope.	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
returning ([meradeemaren, extraatires,])	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text)	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file associated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([normalize_spaces])	Obtain the text (unicode instance)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
(F ) (91)	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for $text()$ with	
\L 3/	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()	
· · · · · · · · · · · · · · · · · ·		

Continued on next page

Table 46 – continued from previous page

xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractTextMarkup'>, <class 'pynlpl.formats.folia.AbstractTex
ANNOTATIONTYPE = 0
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Text'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL\_ATTRIBS = (1, 2, 3, 5, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = True
TEXTDELIMITER = None
XLINK = True
XMLTAG = 't'
Method Details
__init__ (doc, *args, **kwargs)
               Example:
                text = folia.TextContent(doc, 'test')
                text = folia.TextContent(doc, 'test', cls='original')
__init__ (doc, *args, **kwargs)
                Example:
```

```
text = folia.TextContent(doc, 'test')
text = folia.TextContent(doc, 'test', cls='original')
```

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

• **newdoc** (*Document*) – The document the copy should be associated with.

• idsuffix (str or bool) – If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

## findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### finddefaultreference()

Find the default reference for text offsets: The parent of the current textcontent's parent (counting only Structure Elements and Subtoken Annotation Elements)

Note: This returns not a TextContent element, but its parent. Whether the textcontent actually exists is checked later/elsewhere

# classmethod findreplaceables (parent, set, \*\*kwargs)

(Method for internal usage, see AbstractElement)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

## Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### getreference (validate=True)

Returns and validates the Text Content's reference. Raises UnresolvableTextContent when invalid

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

See AbstractElement. json ()

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
classmethod parsexml (node, doc, **kwargs)
```

(Method for internal usage, see AbstractElement)

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

## See also:

```
phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()
```

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\verb|classmethod| relaxing| (include children = True, extraattribs = None, extraelements = None)$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

## settext (text)

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
```

Alias for text () with strict=True

text (normalize\_spaces=False)

Obtain the text (unicode instance)

### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.PhonContent

```
class pynlpl.formats.folia.PhonContent(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Phonetic content element (ph), holds a phonetic representation to be associated with whatever element the phonetic content element is a child of.

Phonetic content elements behave much like text content elements.

Phonetic content elements can specify offset that refer to phonetic content at a higher parent level. Use the following keyword arguments:

- ref=: The instance to point to, this points to the element holding the text content element, not the text content element itself.
- offset=: The offset where this text is found, offsets start at 0

## **Method Summary**

init(doc, *args, **kwargs)	Example.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	

Continued on next page

T		•		
Table 47 –	CONTINUED	trom	nrevious	nage
IUDIC TI	COLITICICA	11 0111	providuo	page

	ed from previous page
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
i inacorrectionnanaring (cis)	by looking in the underlying corrections where it is
	reused
C' - 11 - C 1 + C ()	
finddefaultreference()	Find the default reference for text offsets: The par-
	ent of the current textcontent's parent (counting only
	Structure Elements and Subtoken Annotation Ele-
	ments)
findreplaceables(parent, set, **kwargs)	(Method for internal usage, see AbstractElement)
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>getreference([validate])</pre>	Return and validate the Phonetic Content's reference.
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
v	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
Toome ([roundiems])	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
([attros, recarse, ignorense])	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
	Returns the next element, if it is of the specified type
next([Class, scope, reverse])	and if it does not cross the boundary of the defined
	•
	Scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	(Method for internal usage, see AbstractElement)
phon()	Obtain the actual phonetic representation (uni-
	code/str instance)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
	Continued on next page
	1 3

Table 47 – continued from previous page

relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
(meradeomaren, extractivos,)	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
reprace(cinici, urgs, kwargs)	places any existing child element of the same type	
	and set.	
resolveword(id)		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
set doc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
sceparenes()	scop.	
setphon(phon)	Set the representation for the phonetic content (uni-	
seephon(phon)	code instance), called whenever phon= is passed as a	
	keyword argument to an element constructor	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
opeon_openier()	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text()	

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
```

LABEL = 'Phonetic Content'

OCCURRENCES = 0

OCCURRENCES\_PER\_SET = 0

```
OPTIONAL\_ATTRIBS = (1, 2, 3, 5, 11)
PHONCONTAINER = True
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'ph'
Method Details
__init__(doc, *args, **kwargs)
    Example:
    phon = folia.PhonContent(doc, 'hl')
    phon = folia.PhonContent(doc, 'hl', cls="original")
___init___(doc, *args, **kwargs)
    Example:
    phon = folia.PhonContent(doc, 'hl')
    phon = folia.PhonContent(doc, 'hl', cls="original")
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
        Raises ValueError
addidsuffix (idsuffix, recursive=True)
    Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
    call this directly, invoked implicitly by copy ()
```

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy()* on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### **Returns** str or list

## findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### finddefaultreference()

Find the default reference for text offsets: The parent of the current textcontent's parent (counting only Structure Elements and Subtoken Annotation Elements)

Note: This returns not a TextContent element, but its parent. Whether the textcontent actually exists is checked later/elsewhere

## classmethod findreplaceables (parent, set, \*\*kwargs)

(Method for internal usage, see AbstractElement)

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### getreference (validate=True)

Return and validate the Phonetic Content's reference. Raises UnresolvableTextContent when invalid

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

(Method for internal usage, see AbstractElement)

#### phon()

Obtain the actual phonetic representation (unicode/str instance)

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

### setphon (phon)

Set the representation for the phonetic content (unicode instance), called whenever phon= is passed as a keyword argument to an element constructor

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

• **text** (str) – The text

• **cls** (str) – The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

## See also:

```
AbstractElement.xmlstring() - for direct string output
```

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

### Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...

_len__()
    Returns the number of child elements under the current element.
__str__()
    Alias for text()
```

Text is retrieved as string using AbstractElement.text(), or as element using Phonetic content is retrieved as string using AbstractElement.text(), or as element using AbstractElement.textcontent().

**Note:** These are the only elements for which FoLiA prescribes a default set and a default class (current). This will only be relevant if you work with multiple text layers (current text vs OCRed text for instance) or with corrections of orthography or phonetics.

## **Span Annotation**

FoLiA distinguishes token annotation and span annotation, token annotation is embedded in-line within a structural element, and the annotation therefore pertains to that structural element, whereas span annotation is stored in a stand-off annotation layer outside the element and refers back to it. Span annotation elements typically *span* over multiple structural elements, they are all subclasses of *AbstractSpanAnnotation*.

We will discuss three ways of accessing span annotation. As stated, span annotation is contained within an annotation layer (a subclass of <code>AbstractAnnotationLayer</code>) of a certain structure element, often a sentence. In the first way of accessing span annotation, we do everything explicitly: We first obtain the layer, then iterate over the span annotation elements within that layer, and finally iterate over the words to which the span applies. Assume we have a <code>sentence</code> and we want to print all the named entities in it, assuming the entities layer is embedded at sentence level as is conventional:

```
for layer in sentence.select(folia.EntitiesLayer):
    for entity in layer.select(folia.Entity):
        print(" Entity class=", entity.cls, " words=")
        for word in entity.wrefs():
            print(word, end="") #print without newline
            print() #print newline
```

The AbstractSpanAnnotation.wrefs() method, available on all span annotation elements, will return a list of all words (as well as morphemes and phonemes) over which a span annotation element spans.

This first way is rather verbose. The second way of accessing span annotation takes another approach, using the *Word.findspans()* method available on *Word* instances. Here we start from a word and seek span annotations in which that word occurs. Assume we have a word and want to find chunks it occurs in:

```
for chunk in word.findspans(folia.Chunk):
    print(" Chunk class=", chunk.cls, " words=")
    for word2 in chunk.wrefs(): #print all words in the chunk (of which the word is a
    →part)
        print(word2, end="")
    print()
```

The Word. findspans () method can be called with either the class of a Span Annotation Element, such as Chunk, or with the class of the layer, such as ChunkingLayer.

The third way allows us to look for span elements given an annotation layer and words. In other words, it checks if one or more words form a span. This is an exact match and not a sub-part match as in the previously described method. To do this, we use the <code>AbstractAnnotationLayer.findspan</code> method, available on all annotation layers:

```
for span in annotationlayer.findspan(word1,word2):
    print("Class: ", span.cls)
    print("Text: ", span.text()) #same for every span here
```

## **Span Annotation Types**

This section lists the available Span annotation elements, the layer that contains them is explicitly mentioned as well.

Some of the span annotation elements are complex and take span role elements as children, these are normal span annotation elements that occur on a within another span annotation (of a particular type) and can not be used standalone.

FoLiA distinguishes the following span annotation elements:

Chunk	Chunk element, span annotation element to be used in
	ChunkingLayer
CoreferenceChain	Coreference chain.
Dependency	Span annotation element to encode dependency rela-
	tions
Entity	Entity element, for entities such as named entities,
	multi-word expressions, temporal entities.
Observation	Observation.
Predicate	Predicate, used within SemanticRolesLayer,
	takes SemanticRole annotations as children, but has
	its own annotation type and separate declaration
Sentiment	Sentiment.
Statement	Statement.
SyntacticUnit	Syntactic Unit, span annotation element to be used in
	SyntaxLayer
SemanticRole	Semantic Role
TimeSegment	A time segment

# pynlpl.formats.folia.Chunk

```
class pynlpl.formats.folia.Chunk(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
```

Chunk element, span annotation element to be used in ChunkingLayer

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
	Continued on next page

4.1. Reading FoLiA 515

leftcontext(size[, placeholder, scope])	Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.
<pre>items([founditems])  json([attribs, recurse, ignorelist])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a
insert(index, child, *args, **kwargs)	Paturns a danth first flat list of all items below this
ingont(index child *cree **layeres)	wise it returns None
	the Correction element (evaluating to True), other-
incorrection()	Is this element part of a correction? If it is, it returns
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
	ified class)
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	tation exists, and if so, how many.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
generate_id(cls)	
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
feat(subset)	Obtain the feature class value of the specific subset.
description()	Obtain the description associated with the element.
deepvalidation()	Perform deep validation of this element.
	them.
([,,, -0,])	stead of returning the elements, it merely counts
count(Class[, set, recursive, ignore, node])	written still)  Like AbstractElement.select(), but in-
correct(**kwargs)	element.  Apply a correction (TODO: documentation to be
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
	the current word, and {size} words to the right
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
append(child, *args, **kwargs)	See AbstractElement.append()
annotations(Class[, set])	tiple). Obtain annotations.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
	fectively back-tracing its path to the root element.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
ancestor ("Classes)	type, multiple classes may be specified.
ancestor(*Classes)	properly added to the index  Find the most immediate ancestor of the specified
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	to all child IDs as well.
	Appends a suffix to this element's ID, and optionally

		•		
Table 49 –	AANTINIIAA	trom	nralia	n - n - n
14UIE 45 —	COHIHIDED	11 ( )1 1 1	DIEVIDIES.	Dau:

	ed from previous page
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
- · · ·	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
2 (*	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
, , , , , , , , , , , , , , , , , , ,	scop.
setspan(*args)	Sets the span of the span element anew, erases all
	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
, 3)	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
([,	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
([])	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
wrefs([index, recurse])	Returns a list of word references, these can be Words
= = = ([maon, recarse])	but also Morphemes or Phonemes.
	out also interpretates of I fiolicines.
xm1([attribs elements skinchildren])	See AbstractElement xml()
xml([attribs, elements, skipchildren])	See AbstractElement.xml() Serialises this Fol iA element and all its contents to
<pre>xml([attribs, elements, skipchildren]) xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to

Table 49 – continued from previous page

len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 14
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Chunk'
OCCURRENCES = 0
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'chunk'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
```

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

#### annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*)-

## Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

## copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

## correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

### **count** (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

## findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

## generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

## Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

## **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext(cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

```
remove (child)
```

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

### setspan(\*args)

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text. cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

#### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls(str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

 normalize\_spaces (bool) – Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

#### toktext (cls='current')

Alias for text() with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### wrefs (index=None, recurse=True)

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children **Return type** str

\_\_iter\_\_()

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

\_\_str\_\_()
Alias for text()

## pynlpl.formats.folia.CoreferenceChain

class pynlpl.formats.folia.CoreferenceChain (doc, \*args, \*\*kwargs)

 $Bases: \verb|pynlpl.formats.folia.AbstractSpanAnnotation| \\$ 

Coreference chain. Holds CoreferenceLink instances.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
•	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
	Continued on post page

Continued on next page

T		•		
Table 50 -	CONTINUED	trom	nrevious	nage
iabic oo	COLITICIO	11 0111	picvious	page

Table 50 – continued from previous page				
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-			
	stead of returning the elements, it merely counts			
	them.			
deepvalidation()	Perform deep validation of this element.			
description()	Obtain the description associated with the element.			
feat(subset)	Obtain the feature class value of the specific subset.			
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass			
	by looking in the underlying corrections where it is			
	reused			
findreplaceables(parent[, set])	Internal method to find replaceable elements.			
generate_id(cls)				
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive			
	by default!			
getmetadata([key])	Get the metadata that applies to this element, auto-			
	matically inherited from parent elements			
gettextdelimiter([retaintokenisation])	Return the text delimiter for this class.			
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-			
	tation exists, and if so, how many.			
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)			
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)			
incorrection()	Is this element part of a correction? If it is, it returns			
	the Correction element (evaluating to True), other-			
	wise it returns None			
<pre>insert(index, child, *args, **kwargs)</pre>				
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this			
	element (not limited to AbstractElement)			
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a			
	Python dictionary suitable for serialisation to JSON.			
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.			
next([Class, scope, reverse])	Returns the next element, if it is of the specified type			
	and if it does not cross the boundary of the defined			
	scope.			
originaltext([cls])	Alias for retrieving the original uncorrect text.			
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-			
	ment into an instance of the Class.			
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this			
	element (of the specified class)			
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with			
	this element (of the specified class).			
postappend()	This method will be called after an element is added			
	to another and does some checks.			
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified			
	type and if it does not cross the boundary of the de-			
	fined scope.			
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an			
	XML element (lxml.etree) rather than a string)			
remove(child)	Removes the child element			
replace(child, *args, **kwargs)	Appends a child element like append(), but re-			
	places any existing child element of the same type			
	and set.			
	Continued on next page			

Table 50 – continued from previous page

7 7/10	ara p. o		
resolveword(id)			
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.		
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.		
setdoc(newdoc)	Set a different document.		
setdocument(doc)	Associate a document with this element.		
setparents()	Correct all parent relations for elements within the		
	scop.		
setspan(*args)	Sets the span of the span element anew, erases all		
	data inside.		
settext(text[, cls])	Set the text for this element.		
speech_speaker()	Retrieves the speaker of the audio or video file asso-		
	ciated with the element.		
speech_src()	Retrieves the URL/filename of the audio or video file		
	associated with the element.		
stricttext([cls])	Alias for text() with strict=True		
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-		
	ified class)		
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this		
	element (of the specified class).		
textvalidation([warnonly])	Run text validation on this element.		
toktext([cls])	Alias for text() with		
	retaintokenisation=True		
updatetext()	Recompute textual value based on the text content of		
	the children.		
wrefs([index, recurse])	Returns a list of word references, these can be Words		
	but also Morphemes or Phonemes.		
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()		
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to		
- · • • • • ·	XML.		
iter()	Iterate over all children of this element.		
len()	Returns the number of child elements under the cur-		
	rent element.		
str()	Alias for text ()		

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 28

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Coreference Chain'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False

PRIMARYELEMENT = True
```

PRINTABLE = True

```
REQUIRED ATTRIBS = None
REQUIRED_DATA = (<class 'pynlpl.formats.folia.CoreferenceLink'>,)
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'coreferencechain'
Method Details
init (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
        Raises ValueError
addidsuffix (idsuffix, recursive=True)
    Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
    call this directly, invoked implicitly by copy ()
addtoindex (norecurse=None)
    Makes sure this element (and all subelements), are properly added to the index
ancestor (*Classes)
    Find the most immediate ancestor of the specified type, multiple classes may be specified.
        Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
            from. Not instances!
    Example:
    paragraph = word.ancestor(folia.Paragraph)
```

```
ancestors(Class=None)
```

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*) -

Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### **Returns** str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

### Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

```
hastext (cls='current', strict=True, correctionhandling=1)
```

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

## Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### Parameters

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

# originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if

you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the replaced element will be made into an alternative. Simply use AbstractElement.append() if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

## $\verb"resolveword"\,(id)$

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

## Example:

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

 $\label{text} \begin{tabular}{ll} \textbf{text} (cls='current', retain to ken is at ion=False, previous delimiter=", strict=False, correction handling=1, normalize\_spaces=False) \\ \end{tabular}$ 

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if

you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

See also:

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Dependency

```
class pynlpl.formats.folia.Dependency (doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
```

Span annotation element to encode dependency relations

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are multiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
dependent()	Returns the dependent of the dependency relation.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
	Continued on next page

4.1. Reading FoLiA

<b>T</b>		•		
Iahla 51	<ul><li>continued</li></ul>	trom	nravinie	nage
Table 51	COLLINGCA	11 0111	picvious	page

	ed from previous page
head()	Returns the head of the dependency relation.
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
setspan(*args)	Sets the span of the span element anew, erases all
	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
-	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
/	•
	ified class)
textcontent([cls, correctionhandling])	
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
<pre>textcontent([cls, correctionhandling]) textvalidation([warnonly])</pre>	

Table 51 – continued from previous page

rabio o i contini	aca nem previous page
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
wrefs([index, recurse])	Returns a list of word references, these can be Words
	but also Morphemes or Phonemes.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 22
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Dependency'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = (<class 'pynlpl.formats.folia.DependencyDependent'>, <class 'pynlpl.fo
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'dependency'
Method Details
__init__(doc, *args, **kwargs)
```

4.1. Reading FoLiA

Initialize self. See help(type(self)) for accurate signature.

```
__init__ (doc, *args, **kwargs)
```

Initialize self. See help(type(self)) for accurate signature.

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

#### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

#### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

### annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

# **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve(\*)-

Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### dependent()

Returns the dependent of the dependency relation. Instance of DependencyDependent

#### description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### **Returns** int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

#### head()

Returns the head of the dependency relation. Instance of Headspan

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

### replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

# resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.

• **node** (\*) – Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by <code>copy()</code>

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
speech src()
```

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *text* (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

### **Return type** str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

### len ()

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.Entity

```
class pynlpl.formats.folia.Entity(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
```

Entity element, for entities such as named entities, multi-word expressions, temporal entities. This is a span annotation element to be used in *EntitiesLayer* 

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
	Continued on next page

Continued on next page

Table 52 – continued from previous page

Table 52 – contin	ued from previous page
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
(1)	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
amo oa o z on (cyp*t, 55.1)	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
concent (size), placeholaer, scope))	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
Timacorrectionnanaring(cis)	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	internal method to find replaceable elements.
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
getinaex(ciniat, recursive, ignore))	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
_ (_ / / / / / / / / / / / / / / / / / /	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
items([founditems])	Returns a depth-first flat list of <i>all</i> items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
······································	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
∑ (L3/	Continued on next page
	Outlinded on next page

Table 52 –	continued	from	previous	page

	ued from previous page
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but replaces any existing child element of the same type and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
setspan(*args)	Sets the span of the span element anew, erases all
	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file associated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the specified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
<pre>wrefs([index, recurse])</pre>	Returns a list of word references, these can be Words
	but also Morphemes or Phonemes.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the current element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 15
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Entity'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'entity'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
___init___(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
```

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

#### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

# annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

### **Parameters**

- Class The Class you want to retrieve (\*) -
- set The set you want to retrieve (\*)-

### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

### append (child, \*args, \*\*kwargs)

```
See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str or bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

### copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# $generate\_id(cls)$

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

# Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

#### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

4.1. Reading FoLiA

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

### setspan(\*args)

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

# speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike  $t \in xt$  (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the  $T \in xt$  content instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

#### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

### xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:

__len__()
Returns the number of child elements under the current element.

__str__()
Alias for text()
```

# pynlpl.formats.folia.Observation

```
class pynlpl.formats.folia.Observation(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
    Observation.
```

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are multiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
	Continued on next page

Table 53 –	continued from previous page
$\alpha(cle)$	Find the proper correction

Table 33 – Continu	ed from previous page
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	, 100 10 10 10 10 10 10 10 10 10 10 10 10
items([founditems])	Returns a depth-first flat list of all items below this
([	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
<i>y</i> ( <b>L</b>	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
1	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
<pre>phon([cls, previousdelimiter, strict,]) phoncontent([cls, correctionhandling])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with
<pre>phoncontent([cls, correctionhandling])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).
•	element (of the specified class)  Get the phonetic content explicitly associated with
<pre>phoncontent([cls, correctionhandling])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).
<pre>phoncontent([cls, correctionhandling])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified
<pre>phoncontent([cls, correctionhandling])  postappend()</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
<pre>phoncontent([cls, correctionhandling])  postappend()</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but re-
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but replaces any existing child element of the same type
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)  replace(child, *args, **kwargs)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but replaces any existing child element of the same type
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)  replace(child, *args, **kwargs)  resolveword(id)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but replaces any existing child element of the same type and set.
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)  replace(child, *args, **kwargs)  resolveword(id)  rightcontext(size[, placeholder, scope])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but replaces any existing child element of the same type and set.  Returns the right context for an element, as a list.
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)  replace(child, *args, **kwargs)  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but replaces any existing child element of the same type and set.  Returns the right context for an element, as a list.  Select child elements of the specified class.

Continued on next page

Table 53 – continued from previous page

	u irom previous page	
setparents()	Correct all parent relations for elements within the	
	scop.	
setspan(*args)	Sets the span of the span element anew, erases all	
	data inside.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
wrefs([index, recurse])	Returns a list of word references, these can be Words	
	but also Morphemes or Phonemes.	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text()	

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 43

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Observation'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = True

REQUIRED_ATTRIBS = None

REQUIRED_DATA = None

SETONLY = False
```

```
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'observation'
Method Details
__init__ (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
```

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
\verb"addtoindex" (no recurse = None)"
```

Makes sure this element (and all subelements), are properly added to the index

```
ancestor (*Classes)
```

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

### annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*)-

### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### **Returns** int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class. set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

```
hastext (cls='current', strict=True, correctionhandling=1)
```

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

**phon** (*cls='current'*, *previousdelimiter="*, *strict=False*, *correctionhandling=1*)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

# **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- **text** (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

```
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
iter ()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Predicate

```
class pynlpl.formats.folia.Predicate(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
```

Predicate, used within SemanticRolesLayer, takes SemanticRole annotations as children, but has its own annotation type and separate declaration

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	mindile soil.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addable(parentl, set, raiscenceptions])	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
addid3dilia(dsdilla[, lectisive])	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
addeoindex([noicedise])	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
uncester (Chastes)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
anecocoro([class])	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
annotation(type[, set])	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
context(size[, piaceholder, scope])	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
copyentraren([newdoc, idsumx])	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
Correct(**kwargs)	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
Count (Class), set, recursive, ignore, node])	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
iinacoirectionnanaiing(cis)	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	internal method to find replaceable elements.
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
getindex(child[, recursive, ignore])	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
getmetadata([KCy])	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
nasamiocactom(Ciass[, SEL])	
hasphon([cls, strict, correctionhandling])	tation exists, and if so, how many.  Does this element have phonetic content (of the spec-
masphon([cis, suici, correctionnanding])	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
THEOTTECCTOH	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	wise it feturis from
THOUSE CHICA, CHICA, args, Kwargs)	Continued on next page
	Continued on next page

Table 54 – continued from previous page

Table 54 – continu	ied from previous page
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
(1 / 1 / 1)	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
1	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
priori([eis, previousdeminier, strict,])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
phoneoneene ([cis, correctionnanding])	this element (of the specified class).
postappend()	This method will be called after an element is added
poscappena()	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
previous([Class, scope])	type and if it does not cross the boundary of the de-
	7.2
relaxng([includechildren, extraattribs,])	fined scope.  Returns a RelaxNG definition for this element (as an
reraxing([includecimaten, extraatinos,])	`
(abild)	XML element (lxml.etree) rather than a string)  Removes the child element
remove(child)	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
7 7/ 1\	and set.
resolveword(id)	Determent has right content for an element of a list
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
setspan(*args)	Sets the span of the span element anew, erases all
	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
(5.1,, 4.1,, 4.1,, 1)	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
text([cis, retaintokenisation,])	Get the text associated with this element (of the specified class)
text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])	_
	ified class)
	ified class)  Get the text content explicitly associated with this
<pre>textcontent([cls, correctionhandling]) textvalidation([warnonly])</pre>	ified class)  Get the text content explicitly associated with this element (of the specified class).
textcontent([cls, correctionhandling])	ified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.
<pre>textcontent([cls, correctionhandling])  textvalidation([warnonly])  toktext([cls])</pre>	ified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True
<pre>textcontent([cls, correctionhandling]) textvalidation([warnonly])</pre>	ified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with

Continued on next page

Table 54 – continued from previous page

	1 1 5
wrefs([index, recurse])	Returns a list of word references, these can be Words
	but also Morphemes or Phonemes.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 42
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Predicate'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'predicate'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

## classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

### annotation (type, set=None)

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (  $\star)\,-\,$
- set The set you want to retrieve (\*)-

# Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str or bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## **Returns** str or list

# ${\tt findcorrectionhandling}\,(\mathit{cls})$

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# $\mathtt{generate\_id}\left(\mathit{cls}\right)$

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

## Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
    node - XML Element (*) -
    doc - Document (*) -
```

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.

- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

### setspan(\*args)

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• **cls** (str) – The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

wrefs (index=None, recurse=True)

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

**xml** (attribs=None, elements=None, skipchildren=False)

See AbstractElement.xml()

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
    ...
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Sentiment

```
class pynlpl.formats.folia.Sentiment(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
```

Sentiment. Takes span roles Headspan, Source and Target as children

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on post page

Continued on next page

T 11 FF		•		
Iania sh -	CONTINUIOR	trom	nravialie	naga
Table 55 –	COHUHUCU	HUHH	DIEVIOUS	vauc

Table 55 – continue	ed from previous page
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a single annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
	Continued on next page

Table 55 -	continued	from	previous	page
14010 00	00111111000		picticac	229

	led from previous page
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
J(L	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
repraes(cime, args, kwargs)	places any existing child element of the same type
	and set.
resolveword(id)	and set.
· · ·	Detume the might contact for an element, as a list
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
setspan(*args)	Sets the span of the span element anew, erases all
	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
(1)	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
apaacecene()	the children.
wrefs([index, recurse])	Returns a list of word references, these can be Words
wrers([mack, recurse])	but also Morphemes or Phonemes.
vm7/[attribe alamante elrinahildran])	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()  Socialized this Folia alement and all its contents to
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

ACCEPTED\_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats ANNOTATIONTYPE = 44

```
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Sentiment'
OCCURRENCES = 0
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'sentiment'
Method Details
___init___(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
```

Raises ValueError

#### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

## annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

### annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

## **Parameters**

- Class The Class you want to retrieve(\*)-
- set The set you want to retrieve (\*)-

## Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

## **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

## count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

# hasphon (cls='current', strict=True, correctionhandling=1)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

## **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

## originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

## remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

• alternative (bool) – If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element

• be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- node (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

## setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by <code>copy()</code>

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

```
Parameters *args - Instances of Word, Morpheme or Phoneme
```

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

Returns The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

```
See AbstractElement.xml()
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

## Return type str

```
iter ()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

```
__len__()
Returns the number of child elements under the current element.
__str__()
Alias for text()
```

# pynlpl.formats.folia.Statement

```
class pynlpl.formats.folia.Statement(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
```

 $\textbf{Statement. Takes span roles} \ \textit{Headspan}, \texttt{Source and Relation as children}$ 

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accept s(Class[, raiseexceptions, parentinstance])	minute seri.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addable(parent), set, raiseexceptions)	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
addiabatiff (lasanint, recassive))	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
,	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
	Continued on port page

Continued on next page

Table 56 –	continued	from	previous page
Table 50 -	COLLINGEA	11 0111	previous page

	ied from previous page
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
<pre>hasphon([cls, strict, correctionhandling])</pre>	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
(101	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
7 (1' - 1 1 - 1'11 4 4 - 1 1)	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
(-1.11.1)	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
mana larayandid)	and set.
resolveword(id)	Datums the right contact for an element, as a list
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.  Set a different document.
set document(doc)	Associate a document with this element.
setdocument(doc)	
setparents()	Correct all parent relations for elements within the
cot an an (*arga)	Scop.
setspan(*args)	Sets the span of the span element anew, erases all
settext(text[, cls])	data inside.  Set the text for this element.

Continued on next page

Table 56 – continued from previous page

10.010 0	a nom providad pago	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
wrefs([index, recurse])	Returns a list of word references, these can be Words	
	but also Morphemes or Phonemes.	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text()	

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 45
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Statement'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
```

```
TEXTDELIMITER = None

XLINK = False

XMLTAG = 'statement'
```

#### **Method Details**

```
__init__ (doc, *args, **kwargs)
Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
Initialize self. See help(type(self)) for accurate signature.
```

classmethod accepts(Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

## Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

#### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

```
ancestor(*Classes)
```

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

### annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*) -
- set The set you want to retrieve (\*) -

### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

## append (child, \*args, \*\*kwargs)

See AbstractElement.append()

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str or bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

```
getmetadata (key=None)
```

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is <code>CorrectionHandling.CURRENT</code>, which will retrieve the corrected/current phonetic content. You can set this to <code>CorrectionHandling.ORIGINAL</code> if you want the phonetic content prior to correction, and <code>CorrectionHandling.EITHER</code> if you don't care.

### Returns bool

```
hastext (cls='current', strict=True, correctionhandling=1)
```

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

• **cls** (str) – The class of the text content to obtain, defaults to current.

- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

## originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

Returns An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & orig-class = None) \\ \end{tabular}$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

## Example:

## setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- **text** (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

```
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
iter ()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.SyntacticUnit

```
class pynlpl.formats.folia.SyntacticUnit(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
```

Syntactic Unit, span annotation element to be used in SyntaxLayer

# **Method Summary**

Tests whether a new element of this class can be added to the parent.  Appends a suffix to this element's ID, and optionally to all child IDs as well.
added to the parent.  Appends a suffix to this element's ID, and optionally
added to the parent.  Appends a suffix to this element's ID, and optionally
Appends a suffix to this element's ID, and optionally
Makes sure this element (and all subelements), are properly added to the index
Find the most immediate ancestor of the specified type, multiple classes may be specified.
Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
Will return a <b>single</b> annotation (even if there are multiple).
Obtain annotations.
See AbstractElement.append()
Returns this word in context, {size} words to the left, the current word, and {size} words to the right
Make a deep copy of this element and all its children.
Generator creating a deep copy of the children of this element.
Apply a correction (TODO: documentation to be written still)
Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
Perform deep validation of this element.
Obtain the description associated with the element.
Obtain the feature class value of the specific subset.
Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
Internal method to find replaceable elements.
Get the index at which an element occurs, recursive by default!
Get the metadata that applies to this element, automatically inherited from parent elements
Return the text delimiter for this class.
Returns an integer indicating whether such as annotation exists, and if so, how many.
Does this element have phonetic content (of the specified class)
Does this element have text (of the specified class)
Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

4.1. Reading FoLiA

Table 57 –	continued	from	previous page
Table 37 -	COMMINICA	110111	previous page

	led from previous page
items([founditems])	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
remot([emass, seepe, reverse])	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
parsexiii (node, doc, **kwargs)	ment into an instance of the Class.
. (f.1 1.1'	
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
setspan(*args)	Sets the span of the span element anew, erases all
seespan( <b>mg</b> s)	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
Speech_Speaker()	ciated with the element.
anaah ana()	Retrieves the URL/filename of the audio or video file
speech_src()	
24 24 24 4 2 24 ([a]a])	associated with the element.
stricttext([cls])	Alias for text() with strict=True
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
	Continued on next page

Continued on next page

Table 57 – continued from previous page

Table 67 Continues	a nom providuo pago
wrefs([index, recurse])	Returns a list of word references, these can be Words
	but also Morphemes or Phonemes.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 13
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Syntactic Unit'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'su'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

4.1. Reading FoLiA

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

## classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by COPY()

### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

## annotation (type, set=None)

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (  $\star)\,-\,$
- set The set you want to retrieve (\*)-

## Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## **Returns** str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

## $\mathtt{generate\_id}\left(\mathit{cls}\right)$

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

### json (attribs=None, recurse=True, ignorelist=False)

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

## originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
    node - XML Element (*) -
    doc - Document (*) -
```

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.

- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

## See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• **cls** (str) – The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

wrefs (index=None, recurse=True)

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

See AbstractElement.xml()

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter___()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.SemanticRole

```
class pynlpl.formats.folia.SemanticRole(doc, *args, **kwargs)
```

Bases: pynlpl.formats.folia.AbstractSpanAnnotation

Semantic Role

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on part nage

Continued on next page

Table 58 – continued from previous page

Table 58 – continu	ued from previous page
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a single annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
-	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
-	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	•
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
<b>,</b>	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
- 1 / 1/	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
· · · · · · · · · · · · · · · · · · ·	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
1 - (L, F , 541-4, 1)	element (of the specified class)
	Continued on next page
	Continuou on noxt page

Table 58 – continu	ued from previous page	
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with	
	this element (of the specified class).	
postappend()	This method will be called after an element is added	
	to another and does some checks.	
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified	
	type and if it does not cross the boundary of the de-	
	fined scope.	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within th	
	scop.	
setspan(*args)	Sets the span of the span element anew, erases all	
	data inside.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
wrefs([index, recurse])	Returns a list of word references, these can be Words	
	but also Morphemes or Phonemes.	
xml([attribs, elements, skipchildren])	See AbstractElement.xml()	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	

## **Class Attributes**

\_iter\_\_()

\_str\_\_()

ACCEPTED\_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats ANNOTATIONTYPE = 29

rent element.
Alias for text()

Iterate over all children of this element.

Returns the number of child elements under the cur-

```
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Semantic Role'
OCCURRENCES = 0
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED\_ATTRIBS = (1,)
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'semrole'
Method Details
___init___(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
```

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

## annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

## **Parameters**

- Class The Class you want to retrieve(\*)-
- set The set you want to retrieve (\*)-

## Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

### context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

## **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

## count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

### deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

## hasphon (cls='current', strict=True, correctionhandling=1)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

## originaltext(cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

## remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

• alternative (bool) – If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element

• be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- node (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

## setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

## speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

## **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text() with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

```
See AbstractElement.xml()
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

## **Return type** str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

```
__len__()
Returns the number of child elements under the current element.
__str__()
Alias for text()
```

## pynlpl.formats.folia.TimeSegment

```
class pynlpl.formats.folia.TimeSegment (doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanAnnotation
    A time segment
```

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
	Continued on next page

Table 59 – continued from previous page

	ed from previous page
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
_	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ [ ([ ([ ([ ([ ( ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ( ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ ([ [ ([ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [ [	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
par sermi (node, doe, nwargs)	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
phon([els, previousdenimer, suret,])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
phonocine ([elo; correction and mg])	this element (of the specified class).
postappend()	This method will be called after an element is added
postuppena()	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
previous([elass, scope])	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
Teraxing ([meradeemaren, extraction,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace(clind, args, kwargs)	places any existing child element of the same type
	and set.
resolveword(id)	and 50t.
rightcontext(size[, placeholder, scope])	
	Returns the right context for an element, as a list
salact(Class[ set recursive ignore nodel)	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Select child elements of the specified class. Set a different document.
setdoc(newdoc) setdocument(doc)	Select child elements of the specified class. Set a different document. Associate a document with this element.
setdoc(newdoc)	Select child elements of the specified class.  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the
setdoc(newdoc) setdocument(doc) setparents()	Select child elements of the specified class.  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.
setdoc(newdoc) setdocument(doc)	Select child elements of the specified class.  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.  Sets the span of the span element anew, erases all
setdoc(newdoc) setdocument(doc) setparents()	Select child elements of the specified class.  Set a different document.  Associate a document with this element.  Correct all parent relations for elements within the scop.

Continued on next page

Table 59 – continued from previous page

14515 55 551111145	a nom providuo pago	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
wrefs([index, recurse])	Returns a list of word references, these can be Words	
	but also Morphemes or Phonemes.	
xml([attribs, elements, skipchildren])	See AbstractElement.xml()	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text ()	

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.ActorFeature'>, <class 'pynlpl.formats.f
ANNOTATIONTYPE = 23
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Time Segment'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
```

```
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'timesegment'
```

#### **Method Details**

```
__init__ (doc, *args, **kwargs)
Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
Initialize self. See help(type(self)) for accurate signature.
```

**classmethod accepts** (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

## Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

#### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

```
ancestor (*Classes)
```

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation(type, set=None)
```

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*) -

### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

## generate\_id(cls)

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

```
getmetadata (key=None)
```

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is <code>CorrectionHandling.CURRENT</code>, which will retrieve the corrected/current phonetic content. You can set this to <code>CorrectionHandling.ORIGINAL</code> if you want the phonetic content prior to correction, and <code>CorrectionHandling.EITHER</code> if you don't care.

## Returns bool

```
hastext (cls='current', strict=True, correctionhandling=1)
```

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

• **cls** (str) – The class of the text content to obtain, defaults to current.

- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
ison (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

## Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

## originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

Returns An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

word.phon()

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

## See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

## **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text(str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

## wrefs (index=None, recurse=True)

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

See AbstractElement.xml()

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

## Return type str

```
iter ()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

### \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

These are placed in the following annotation layers:

ChunkingLayer	Chunking Layer: Annotation layer for Chunk span an-
	notation elements
CoreferenceLayer	Syntax Layer: Annotation layer for SyntacticUnit
	span annotation elements
DependenciesLayer	Dependencies Layer: Annotation layer for
	Dependency span annotation elements.
EntitiesLayer	Entities Layer: Annotation layer for Entity span an-
	notation elements.
	Continued on next page

Table 60 -	continued	from	previous page
Table 00 -	COMMINICA	11 0111	previous page

ObservationLayer	Observation Layer: Annotation layer for
	Observation span annotation elements.
SentimentLayer	Sentiment Layer: Annotation layer for Sentiment
	span annotation elements, used for sentiment analysis.
StatementLayer	Statement Layer: Annotation layer for Statement
	span annotation elements, used for attribution annota-
	tion.
SyntaxLayer	Syntax Layer: Annotation layer for SyntacticUnit
	span annotation elements
SemanticRolesLayer	Syntax Layer: Annotation layer for SemanticRole
	span annotation elements
TimingLayer	Timing layer: Annotation layer for TimeSegment
	span annotation elements.

# pynlpl.formats.folia.ChunkingLayer

class pynlpl.formats.folia.ChunkingLayer (doc, \*args, \*\*kwargs)

Bases: pynlpl.formats.folia.AbstractAnnotationLayer

Chunking Layer: Annotation layer for Chunk span annotation elements

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
	Continued on next page

Table 61 – continued from previous page

	ued from previous page
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec-
	ified words or morphemes.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
(1.1)	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
7 (6.1 1.1 1.2	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
about a state of follows and the state of th	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
nogt appoint()	this element (of the specified class).  This method will be called after an element is added
postappend()	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
prevrous([Ciass, scope])	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
= = =g ([g, ([])	XML element (lxml.etree) rather than a string)
	Continued on next page
	Strained on now page

Table 61 – continued from previous page

	aca from previous page
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
•	rent element.
str()	Alias for text ()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Chunk'>, <class 'pynlpl.formats.folia.Co.
ANNOTATIONTYPE = 14

AUTH = True

AUTO_GENERATE_ID = False

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)

PHONCONTAINER = False

PRIMARYELEMENT = False

PRINTABLE = False

REQUIRED_ATTRIBS = None
```

```
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'chunking'
```

### **Method Details**

**classmethod addable** (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

# **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

# Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# $\verb|alternatives| (Class=None, set=None)|$

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

# **Parameters**

- Class The Class you want to retrieve (\*) -
- set The set you want to retrieve (  $\star)\,-\,$

**Returns** Generator over Alternative elements

```
ancestor(*Classes)
```

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

```
annotations (Class, set=None)
```

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve (\*)-

# Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
    See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

 $Like \ {\tt AbstractElement.select()}, \ but \ instead \ of \ returning \ the \ elements, \ it \ merely \ counts \ them.$ 

## Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

# findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

## findspan(\*words)

Returns the span element which spans over the specified words or morphemes.

## See also:

```
Word.findspans()
```

```
generate\_id(cls)
```

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

# hasphon (cls='current', strict=True, correctionhandling=1)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

## Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

# originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use AbstractElement.append() if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

**Return type** str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
___len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.CoreferenceLayer

class pynlpl.formats.folia.CoreferenceLayer(doc, \*args, \*\*kwargs)

Bases: pynlpl.formats.folia.AbstractAnnotationLayer

Syntax Layer: Annotation layer for SyntacticUnit span annotation elements

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
• -	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
,	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec-
	ified words or morphemes.
generate_id(cls)	-
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
	Continued on next page
	- 13-

	Table 62 –	continued	from	previous	page
--	------------	-----------	------	----------	------

Get the metadata that applies to this element, automatically inherited from parent elements  gettextdelimiter([retaintokenisation]) hasannotation(Class], set)  Returns an integer indicating whether such as annotation exists, and if so, how many. hasphon([cls, strict, correctionhandling])  Does this element have phonetic content (of the specified class) hastext([cls, strict, correctionhandling])  Incorrection()  Is this element have text (of the specified class) Incorrection()  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Insert(index, child, *args, **kwargs)  Items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Serialises the Folia element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element (of the specified class).  Phon([cls, previousdelimiter, strict,])  Phonocontent([cls, correctionhandling])  Get the phonetic representation associated with this element (of the specified class).  Previous([Class, scope])  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified vipe and if it does not cross the boundary of the defined scope.  Returns the previous element, if it is of the specified vipe and if it does not cross the boundary of the defined scope.  Returns the right context for an element is added to another and does some checks.  Returns the right context for an element, as an ist.  Returns the right context for an element, as an ist.  Returns the right context for an element, as an ist.  Returns the right context		ued from previous page
gettextdel_imitex([retaintokenisation])         Returns an integer indicating whether such as annotation exists, and if so, how many.           hasannotation(Class[, set])         Returns an integer indicating whether such as annotation exists, and if so, how many.           hasphon([cls, strict, correctionhandling])         Does this element have phonetic content (of the specified class)           hastext([cls, strict, correctionhandling])         Does this element have text (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns to the correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           leftcontext(sizel, placeholder, scope)         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the left context for an element, as a list.           parsexml(node, doc, **kwargs)         Alias for retrieving the original uncorrect text.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoneontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           previous([C	getmetadata([key])	Get the metadata that applies to this element, auto-
Returns an integer indicating whether such as annotation exists, and if so, how many.		
tation exists, and if so, how many.  hasphon([cls, strict, correctionhandling])  Does this element have phonetic content (of the specified class)  hastext([cls, strict, correctionhandling])  Incorrection()  Is this element have text (of the specified class)  incorrection()  Is this element have text (of the specified class)  incorrection()  Is this element have text (of the specified class)  incorrection()  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Ieftcontext(size[, placeholder, scope])  next([Class, scope, reverse])  Returns the left context for an element, as a list.  next([cls])  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element (not described class).  phon([cls, previousdelimiter, strict,])  phonocontent([cls, correctionhandling])  phonocontent([cls, correctionhandling])  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified class)  premove(child)  Removes the child element  replace(child, *args, **kwargs)  Returns the previous element, if it is of the specified class)  select(Class, secpel)  Returns the previous element, if it is of the specified class on the construction and the string of the specified class.  Returns the previous element, if it is of the specified class on the selection of the specified class on the selection of the specified class of the specified class.  Returns the previous element, if it is of the specified class of the specified class of the specified class of the specified class.  Returns the previous element, if it is of the specified class of t	<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
Does this element have phonetic content (of the specified class)   Incorrection()	hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
hastext([cls, strict, correctionhandling])         Does this element have text (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **ekwargs)         **ekums a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           leftcontext(size], placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the left context for an element, as a list.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class).           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxing([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (smllettre		tation exists, and if so, how many.
Does this element have text (of the specified class)   incorrection()	hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Period (citionary suitable for serialisation to JSON).  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element (ind in instance of the Class.)  phon([cls, previousdelimiter, strict,])  Get the phonetic content explicitly associated with this element (of the specified class)  phonocontent([cls, correctionhandling])  phonocontent([cls, correctionhandling])  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (kml.etree) rather than a string)  remove(child)  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  setzieves the URL/filename of the audio or video file associated with the element.		ified class)
Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Period (citionary suitable for serialisation to JSON).  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element (ind in instance of the Class.)  phon([cls, previousdelimiter, strict,])  Get the phonetic content explicitly associated with this element (of the specified class)  phonocontent([cls, correctionhandling])  phonocontent([cls, correctionhandling])  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (kml.etree) rather than a string)  remove(child)  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdoc(newdoc)  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  setzieves the URL/filename of the audio or video file associated with the element.	hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Returns the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  next([Class, scope, reverse])  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  parsexm1(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified to another and does some checks.  previous([Class, scope])  Returns a RelaxNG definition for this element (as an XML element (xml etree) rather than a string)  remove(child)  replace(child, *args, **kwargs)  Returns a RelaxNG definition for this element (as an XML element (xml etree) rather than a string)  remove(child)  replace(child, *args, **kwargs)  Previous([Class, set, recursive, ignore, node])  setdoc(newdoe)  setdoc(newdoe)  Set a different document.  setparents()  Set the text for this element.  setparents()  Retrieves the speaker of the audio or video file associated with the element.  setparents()  Retrieves the previous that the element.  setparents()  Retrieves the URL/filename of the audio or video file associated with the element.		
insert(index, child, *args, **kwargs)         wise it returns None           items([founditems])         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           leftcontext(size[, placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           originaltext([cls])         Alias for retrieving the original uncorrect text.           parsexmI(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phonocontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxing([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)	,	
Returns a depth-first flat list of all items below this element (not limited to AbstractElement)		
Returns a depth-first flat list of all items below this element (not limited to AbstractElement)	insert(index. child. *args. **kwargs)	
element (not limited to AbstractElement)   json([attribs, recurse, ignorelist])		Returns a depth-first flat list of <i>all</i> items below this
Jeson([attribs, recurse, ignorelist])   Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.     Returns the left context for an element, as a list.     Returns the left context for an element, as a list.     Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.     Alias for retrieving the original uncorrect text.     Internal class method used for turning an XML element into an instance of the Class.     Phon([cls, previous delimiter, strict,])   Get the phonetic representation associated with this element (of the specified class)     Phoneontent([cls, correction handling])   Get the phonetic content explicitly associated with this element (of the specified class)     Postappend()   This method will be called after an element is added to another and does some checks.     Previous([Class, scope])   Returns the previous element, if it is of the specified to another and does some checks.     Previous([include children, extraattribs,])   Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)     Removes the child element   Removes the child element   Removes the child element   Removes the child element of the same type and set.	_ come([roundrems])	
Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope]) Returns the left context for an element, as a list.  next([Class, scope, reverse]) Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls]) Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,]) Get the phonetic representation associated with this element (of the specified class)  phonocontent([cls, correctionhandling]) Get the phonetic content explicitly associated with this element (of the specified class).  postappend() This method will be called after an element is added to another and does some checks.  previous([Class, scope]) Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,]) Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child) Removes the child element (lxml.etree) rather than a string)  remove(child) Removes the child element like append(), but replaces any existing child element of the same type and set.  resolveword(id) rightcontext(size[, placeholder, scope]) Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node]) Select child element of the specified class.  setdoc(mewdoc) Set a different document.  setdocument(doc) Set a different document.  correct all parent relations for elements within the scop.  setdocument(doc) Set the text for this element.  setereves the speaker of the audio or video file associated with the element.  setereves the speaker of the audio or video file associated with the element.	ison([attribs recurse ignorelist])	
Returns the left context for an element, as a list.   next([Class, scope, reverse])   Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.   originaltext([cls])   Alias for retrieving the original uncorrect text.     parsexml(node, doc, **kwargs)   Internal class method used for turning an XML element into an instance of the Class.     phon([cls, previousdelimiter, strict,])   Get the phonetic representation associated with this element (of the specified class)     phoncontent([cls, correctionhandling])   Get the phonetic content explicitly associated with this element (of the specified class).     postappend()   This method will be called after an element is added to another and does some checks.     previous([Class, scope])   Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.     relaxng([includechildren, extraattribs,])   Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)     remove(child)   Removes the child element     replace(child, *args, **kwargs)   Appends a child element like append(), but replaces any existing child element of the same type and set.     resolveword(id)   rightcontext(size[, placeholder, scope])   Returns the right context for an element, as a list.     select(Class[, set, recursive, ignore, node])   Select child elements of the specified class.     setdocument(doc)   Associate a document with this element.     setdocument(doc)   Associate a document with this element.     settext([ext], cls])   Set the text for this element.     setzect(Liss[)   Retrieves the URL/filename of the audio or video file associated with the element.	JSON([attribs, rectase, ignorenst])	
next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           originaltext([cls])         Alias for retrieving the original uncorrect text.           phon([cls, previousdelimiter, strict,])         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         Returns the right context for an element, as a list.           select(Class[, set, recursive, ignore, node])         Select child elements of the specified class.           setdocument(doc)         Associate a docume	loft contoxt(size[ placeholder scope])	
and if it does not cross the boundary of the defined scope.  originaltext([cls])  parsexml(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])  phoncontent([cls, correctionhandling])  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Removes the child element  replace(child, *args, **kwargs)  place any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(class[, set, recursive, ignore, node])  setdocument(doc)  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  setparents()  Retrieves the URL/filename of the audio or video file associated with the element.  stricttext([cls])  Alias for text () with strict=True		·
originaltext([cls])         Alias for retrieving the original uncorrect text.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxing([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         *** rightcontext(size[, placeholder, scope])         Returns the right context for an element, as a list.           select(Class], set, recursive, ignore, node])         Select child elements of the specified class.           setdocument(doc)         Associate a document with this element.           settext(text[, cls])         Set the text for this element.	HEAL([Class, scope, reverse])	
originaltext([cls])         Alias for retrieving the original uncorrect text.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxing([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         rightcontext(size[, placeholder, scope])         Returns the right context for an element, as a list.           select(Class[, set, recursive, ignore, node])         Select child elements of the specified class.           setdoc(newdoe)         Set a different document.           setparents()         Correct all parent relations for elements within the scop.		-
parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         Returns the right context for an element, as a list.           select(Class[, set, recursive, ignore, node])         Select child elements of the specified class.           setdocument(doc)         Associate a document with this element.           settext(text[, cls])         Set the text for this element.           speech_speaker()         Retrieves the speaker of the audio or video file associated with the element.           speech_src()		±
ment into an instance of the Class.  phon([cls, previousdelimiter, strict,]) Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling]) Get the phonetic content explicitly associated with this element (of the specified class).  postappend() This method will be called after an element is added to another and does some checks.  previous([Class, scope]) Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,]) Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child) Removes the child element  replace(child, *args, **kwargs) Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope]) Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node]) Select child elements of the specified class.  setdoc(newdoc) Set a different document.  setdocument(doc) Associate a document with this element.  setparents() Correct all parent relations for elements within the scop.  settext(text[, cls]) Set the text for this element.  speech_speaker() Retrieves the speaker of the audio or video file associated with the element.  speech_speaker() Retrieves the URL/filename of the audio or video file associated with the element.		
phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.           relaxng([includechildren, extraattribs,])         Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)           remove(child)         Removes the child element           replace(child, *args, **kwargs)         Appends a child element like append(), but replaces any existing child element of the same type and set.           resolveword(id)         resolveword(id)           rightcontext(size[, placeholder, scope])         Returns the right context for an element, as a list.           select(Class[, set, recursive, ignore, node])         Select child elements of the specified class.           setdoc(newdoc)         Set a different document.           setdocument(doc)         Associate a document with this element.           setparents()         Correct all parent relations for elements within the scop.           settext(text[, cls])         Set the text for this element.           speech_speake	parsexm1(node, doc, **kwargs)	
element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  Appends a child element like append(), but replace any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  select(Class[, set, recursive, ignore, node])  setdoc(newdoc)  setdoc(newdoc)  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  speech_speaker()  Retrieves the speaker of the audio or video file associated with the element.  speech_src()  Retrieves the URL/filename of the audio or video file associated with the element.		
phoncontent([cls, correctionhandling])Get the phonetic content explicitly associated with this element (of the specified class).postappend()This method will be called after an element is added to another and does some checks.previous([Class, scope])Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.relaxng([includechildren, extraattribs,])Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)Featurns the right context for an element, as a list.resolveword(ids)Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text () with strict=True	phon([cls, previousdelimiter, strict,])	
this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node])  Select child elements of the specified class.  setdoc(newdoe)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  speech_speaker()  Retrieves the speaker of the audio or video file associated with the element.  speech_src()  Retrieves the URL/filename of the audio or video file associated with the element.		<u> </u>
postappend()This method will be called after an element is added to another and does some checks.previous([Class, scope])Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.relaxng([includechildren, extraattribs,])Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdocument(doc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text () with strict=True	phoncontent([cls, correctionhandling])	
to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node])  Select child elements of the specified class.  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  speech_speaker()  Retrieves the speaker of the audio or video file associated with the element.  speech_src()  Retrieves the URL/filename of the audio or video file associated with the element.  stricttext([cls])  Alias for text() with strict=True		
Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node])  Select child elements of the specified class.  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  speech_speaker()  Retrieves the speaker of the audio or video file associated with the element.  speech_src()  Retrieves the URL/filename of the audio or video file associated with the element.  stricttext([cls])  Alias for text() with strict=True	postappend()	This method will be called after an element is added
type and if it does not cross the boundary of the defined scope.  relaxng([includechildren, extraattribs,])  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  remove(child)  Removes the child element  replace(child, *args, **kwargs)  Appends a child element like append(), but replaces any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node])  Select child elements of the specified class.  setdoc(newdoc)  Set a different document.  setdocument(doc)  Associate a document with this element.  setparents()  Correct all parent relations for elements within the scop.  settext(text[, cls])  Set the text for this element.  speech_speaker()  Retrieves the speaker of the audio or video file associated with the element.  speech_src()  Retrieves the URL/filename of the audio or video file associated with the element.  stricttext([cls])  Alias for text() with strict=True		
fined scope.relaxng([includechildren, extraattribs,])Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)Returns the right context for an element, as a list.rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True	<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
relaxng([includechildren, extraattribs,])Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)Returns the right context for an element, as a list.rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		type and if it does not cross the boundary of the de-
remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)right context (size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.set document(doc)Set a different document.set parents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		fined scope.
remove(child)Removes the child elementreplace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)**rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True	relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
replace(child, *args, **kwargs)Appends a child element like append(), but replaces any existing child element of the same type and set.resolveword(id)Returns the right context for an element, as a list.rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		XML element (lxml.etree) rather than a string)
places any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope]) Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node]) Select child elements of the specified class.  setdoc(newdoc) Set a different document.  setdocument(doc) Associate a document with this element.  setparents() Correct all parent relations for elements within the scop.  settext(text[, cls]) Set the text for this element.  speech_speaker() Retrieves the speaker of the audio or video file associated with the element.  speech_src() Retrieves the URL/filename of the audio or video file associated with the element.  stricttext([cls]) Alias for text() with strict=True		Removes the child element
places any existing child element of the same type and set.  resolveword(id)  rightcontext(size[, placeholder, scope]) Returns the right context for an element, as a list.  select(Class[, set, recursive, ignore, node]) Select child elements of the specified class.  setdoc(newdoc) Set a different document.  setdocument(doc) Associate a document with this element.  setparents() Correct all parent relations for elements within the scop.  settext(text[, cls]) Set the text for this element.  speech_speaker() Retrieves the speaker of the audio or video file associated with the element.  speech_src() Retrieves the URL/filename of the audio or video file associated with the element.  stricttext([cls]) Alias for text() with strict=True	replace(child, *args, **kwargs)	Appends a child element like append(), but re-
and set.resolveword(id)rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		places any existing child element of the same type
rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		
rightcontext(size[, placeholder, scope])Returns the right context for an element, as a list.select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True	resolveword(id)	
select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		Returns the right context for an element, as a list.
setdoc(newdoc)Set a different document.setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		
setdocument(doc)Associate a document with this element.setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		
setparents()Correct all parent relations for elements within the scop.settext(text[, cls])Set the text for this element.speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True		
$ \begin{array}{c} scop. \\ \hline settext(text[,cls]) & Set the text for this element. \\ \hline speech\_speaker() & Retrieves the speaker of the audio or video file associated with the element. \\ \hline speech\_src() & Retrieves the URL/filename of the audio or video file associated with the element. \\ \hline stricttext([cls]) & Alias for text() with strict=True \\ \hline \end{array} $		
settext(text[, cls])       Set the text for this element.         speech_speaker()       Retrieves the speaker of the audio or video file associated with the element.         speech_src()       Retrieves the URL/filename of the audio or video file associated with the element.         stricttext([cls])       Alias for text () with strict=True	ocepatenes()	_
speech_speaker()Retrieves the speaker of the audio or video file associated with the element.speech_src()Retrieves the URL/filename of the audio or video file associated with the element.stricttext([cls])Alias for text() with strict=True	cottoyt(text[ cls])	•
$\begin{array}{c} \text{ciated with the element.} \\ speech\_src() & \text{Retrieves the URL/filename of the audio or video file} \\ \text{associated with the element.} \\ \text{\textit{stricttext([cls])}} & \text{Alias for } \textit{text()} \text{ with } \textit{strict=True} \end{array}$		
	speecii_speakei()	
associated with the element.  stricttext([cls]) Alias for text() with strict=True		
stricttext([cls]) Alias for text() with strict=True	speecn_src()	
	([13]	
Continued on next page	stricttext([cls])	
		Continued on next page

Table 62 – continued from previous page

	1 1 0
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 28
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL\_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'coreferences'
```

# **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*) -

**Returns** Generator over Alternative elements

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

### annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

## **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*)-

## Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

# Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### **Returns** str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

### See also:

```
Word.findspans()
```

```
generate_id(cls)
```

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to

CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

- node XML Element (\*) doc Document (\*) -
- Returns An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

 $phoncontent(): \ \, \text{Retrieves the phonetic content as an element rather than a string } \, \, text() \\ textcontent()$ 

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

# remove(child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use <code>AbstractElement.append()</code> if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*) Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

## setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the

corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

```
Alias for text () with retain to kenisation=True
```

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.DependenciesLayer

```
class pynlpl.formats.folia.DependenciesLayer (doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractAnnotationLayer
```

Dependencies Layer: Annotation layer for Dependency span annotation elements. For dependency entities.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a single annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec-
	ified words or morphemes.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
<pre>getmetadata([key])</pre>	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
<pre>hasphon([cls, strict, correctionhandling])</pre>	Does this element have phonetic content (of the spec-
	ified class)

Table 63 – continued from previous page

	led from previous page
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
_	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
, ,	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
concepto, retaintenementon,])	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
concomentations, concommunity)	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for $text()$ with
	retaintokenisation=True
	Continued on next page
	John Har Daye

Continued on next page

Table 63 – continued from previous page

rable do Continuada nom providuo pago	
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 22
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'dependencies'
Method Details
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

## classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

## **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*)-

**Returns** Generator over Alternative elements

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

# annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

```
• Class - The Class you want to retrieve (*)-
```

• set - The set you want to retrieve (\*)-

Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

## findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

#### See also:

```
Word.findspans()
```

# generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

# hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling – Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

# originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

## See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

 $\label{text} \begin{tabular}{ll} \textbf{text} (cls='current', retain to ken is at ion=False, previous delimiter=", strict=False, correction handling=1, normalize\_spaces=False) \\ \end{tabular}$ 

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

## **Parameters**

- ${\tt cls}\,({\it str})$  The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

text() phoncontent() phon()

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

## Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
```

# \_\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.EntitiesLayer

```
class pynlpl.formats.folia.EntitiesLayer(doc, *args, **kwargs)
Bases: pynlpl.formats.folia.AbstractAnnotationLayer
```

Entities Layer: Annotation layer for Entity span annotation elements. For named entities.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
	0 1 1

Continued on next page

gettextdelimiter([retaintokenisation])

hasphon([cls, strict, correctionhandling])

hastext([cls, strict, correctionhandling])

insert(index, child, \*args, \*\*kwargs)

hasannotation(Class[, set])

incorrection()

items([founditems])

addidsuffix(idsuffix[, recursive])	nued from previous page  Appends a suffix to this element's ID, and optionally
addid at the (lasaling, recalls (e.g.)	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
([101004110])	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of
([	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
(	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef
,	fectively back-tracing its path to the root element.
<pre>annotation(type[, set])</pre>	Will return a <b>single</b> annotation (even if there are mul
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left
• • •	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of thi
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in
	stead of returning the elements, it merely count
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclas
	by looking in the underlying corrections where it i
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec
- '	ified words or morphemes.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto
	matically inherited from parent elements

Continued on next page

Chapter 4. FoLiA library

Return the text delimiter for this class.

tation exists, and if so, how many.

ified class)

wise it returns None

Returns an integer indicating whether such as anno-

Does this element have phonetic content (of the spec-

Does this element have text (of the specified class)

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other-

Returns a depth-first flat list of all items below this

element (not limited to AbstractElement)

10.0.0 0 1 00.1	aca nem premede page
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.

Set a different document.

Set the text for this element.

associated with the element.

element (of the specified class).

Run text validation on this element.

for

retaintokenisation=True

See AbstractElement.xml()

ciated with the element.

ified class)

the children.

Alias

XML.

Associate a document with this element.

Alias for text () with strict=True

Correct all parent relations for elements within the

Retrieves the speaker of the audio or video file asso-

Retrieves the URL/filename of the audio or video file

Get the text associated with this element (of the spec-

Get the text content explicitly associated with this

Recompute textual value based on the text content of

Serialises this FoLiA element and all its contents to

Table 64 – continued from previous page

Iterate over all children of this element.

Continued on next page

setdoc(newdoc)

setparents()

speech\_src()

toktext([cls])

updatetext()

\_iter\_\_()

setdocument(doc)

settext(text[, cls])

speech speaker()

stricttext([cls])

text([cls, retaintokenisation, ...])

textvalidation([warnonly])

xm1([attribs, elements, skipchildren])

xmlstring([pretty\_print])

textcontent([cls, correctionhandling])

with

Table 64 – continued from previous page

len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()

#### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 15
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'entities'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
```

**Parameters** 

by subclasses for more customised behaviour.

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

#### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

#### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve (  $\star)\,-\,$

**Returns** Generator over Alternative elements

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

## **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve (\*) -

#### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
See AbstractElement.append()
```

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

#### findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

#### See also:

```
Word.findspans()
```

#### generate id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation(Class. set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

#### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

## hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
• node - XML Element (*) -
```

```
• doc - Document (*) -
```

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use AbstractElement.append() if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

#### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

#### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

See AbstractElement.xml()

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

\_\_str\_\_()
Alias for text()

## pynlpl.formats.folia.ObservationLayer

```
class pynlpl.formats.folia.ObservationLayer(doc, *args, **kwargs)
```

Bases: pynlpl.formats.folia.AbstractAnnotationLayer

Observation Layer: Annotation layer for Observation span annotation elements.

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
	Continued on next page

4.1. Reading FoLiA

Table 65 -	continued	from	previous page
1able 65 –	Continuea	110111	previous page

	ued from previous page
<pre>annotation(type[, set])</pre>	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec-
	ified words or morphemes.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
·	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
7	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
/ / / / / / / / / / / / / / / / / / /	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
	Continued on next page

Table 65 -	continued	from	previous	page

	ed from previous page
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
21 22 ( 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,	places any existing child element of the same type
	and set.
resolveword(id)	** * *********************************
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
Speecii_Speakei()	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
, ,,	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
([,	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for $text()$ with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
apaacecene()	the children.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
Amiro er ring ([piemy_print])	XML.
	Iterate over all children of this element.
iter()	Returns the number of child elements under the cur-
len()	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 43
AUTH = True
```

OCCURRENCES = 0

AUTO\_GENERATE\_ID = False

```
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = False
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'observations'
Method Details
init (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
        Raises ValueError
addidsuffix (idsuffix, recursive=True)
    Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
    call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
    Makes sure this element (and all subelements), are properly added to the index.
    Mostly for internal use.
```

#### alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*)-

**Returns** Generator over Alternative elements

#### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

#### annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (  $\star$ ) -

## Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

## context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

## count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

#### findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

### See also:

```
Word.findspans()
```

## generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### **Returns** int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

## **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

```
remove (child)
```

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy* ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

Example:

```
word.text()
```

Returns The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

#### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

## updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

**Return type** str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...

_len__()
    Returns the number of child elements under the current element.
__str__()
    Alias for text()
```

## pynlpl.formats.folia.SentimentLayer

```
class pynlpl.formats.folia.SentimentLayer(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractAnnotationLayer
```

Sentiment Layer: Annotation layer for Sentiment span annotation elements, used for sentiment analysis.

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
<pre>annotation(type[, set])</pre>	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
	Continued on next page

4.1. Reading FoLiA

T 1 1 00		•		
Table 66 –	CONTINUED	trom	nrevious	nage
iabic co	COLITICIO	11 0111	picvious	page

	ded from previous page
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec-
	ified words or morphemes.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
<u> </u>	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
5 (L. 1911)	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
	una got.
recoluenced(id)	
resolveword(id)	Paturns the right context for an alamant, as a list
resolveword(id) rightcontext(size[, placeholder, scope]) select(Class[, set, recursive, ignore, node])	Returns the right context for an element, as a list. Select child elements of the specified class.

Table 66 – continued from previous page

setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str ()	Alias for text ()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 44

AUTH = True

AUTO_GENERATE_ID = False

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)

PHONCONTAINER = False

PRIMARYELEMENT = False

PRINTABLE = False

REQUIRED_ATTRIBS = None

REQUIRED_DATA = None

SETONLY = True

SPEAKABLE = False

SUBSET = None
```

```
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'sentiments'
```

#### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve (\*)-

**Returns** Generator over Alternative elements

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

```
annotations (Class, set=None)
```

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*) -
- set The set you want to retrieve  $(\star)$  -

#### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### **Returns** int

## deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

#### findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

#### See also:

```
Word.findspans()
```

## generate\_id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

## Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling — Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

Class (\*) – The class to select; any python class subclassed off 'AbstractElement', may
also be a tuple of multiple classes. Set to True to constrain to the same class as that of
the current instance, set to None to not constrain at all

• **scope** (\*) – A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

#### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correction handling = Correction Handling . ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### Parameters

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

## See also:

```
phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()
```

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- alternative (bool) If set to True, the replaced element will be made into an alternative. Simply use AbstractElement.append() if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

## setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

• correctionhandling – Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

#### Returns bool

```
toktext (cls='current')
```

```
Alias for text () with retain token is at ion=True
```

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

```
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

## \_\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.StatementLayer

```
class pynlpl.formats.folia.StatementLayer(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractAnnotationLayer
```

Statement Layer: Annotation layer for Statement span annotation elements, used for attribution annotation.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are multiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the specified words or morphemes.
generate_id(cls)	-
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.

4.1. Reading FoLiA

	ed from previous page
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
- · · · · · · · · · · · · · · · · · · ·	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
<del></del>	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
<del></del> -	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
	Continued on next page

Continued on next page

Table 67 – continued from previous page

	1 1 5
updatetext()	Recompute textual value based on the text content of
	the children.
xml([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 45
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'statements'
Method Details
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

4.1. Reading FoLiA 717

### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*) -

**Returns** Generator over Alternative elements

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

# annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

```
• Class - The Class you want to retrieve (*)-
```

• set - The set you want to retrieve (\*)-

Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

# ${\tt deepvalidation}\,(\,)$

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

# findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

#### See also:

```
Word.findspans()
```

# generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

#### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

# hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling – Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

# originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

# previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

# speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

text() phoncontent() phon()

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

```
\textbf{xml} \; (attribs = None, \; elements = None, \; skipchildren = False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
    ...
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.SyntaxLayer

```
class pynlpl.formats.folia.SyntaxLayer(doc, *args, **kwargs)
```

Bases: pynlpl.formats.folia.AbstractAnnotationLayer

Syntax Layer: Annotation layer for SyntacticUnit span annotation elements

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.

Continued on next page

Table 68 – continued from previous page

	ied from previous page
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are multiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
conv([newdoc_ideuffix])	Make a deep copy of this element and all its children.
<pre>copy([newdoc, idsuffix]) copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the specified words or morphemes.
generate_id(cls)	•
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
mach, chira, argo, margo)	

Table 68 – continu	ued from previous page
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
(111)	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
[	this element (of the specified class).
postappend()	This method will be called after an element is added
postappena()	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
provide ([Ciuss, scope])	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
reraxing ([includeennaren, extraatirios,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace(cliffd, raigs, rewaigs)	places any existing child element of the same type
	and set.
resolveword(id)	and set.
rightcontext(size[, placeholder, scope])	Datums the right contact for an element, as a list
	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.  Set a different document.
setdoc(newdoc)	
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
(4.45.41.1)	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
7	
speech_src()	Retrieves the URL/filename of the audio or video file
	Retrieves the URL/filename of the audio or video file associated with the element.
stricttext([cls])	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True
	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the spec-
<pre>stricttext([cls]) text([cls, retaintokenisation,])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)
stricttext([cls])	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  See AbstractElement.xml()
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.
<pre>stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()  xml([attribs, elements, skipchildren])</pre>	Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  See AbstractElement.xml()  Serialises this FoLiA element and all its contents to

Continued on next page

Table 68 – continued from previous page

len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 13
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'syntax'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
```

by subclasses for more customised behaviour.

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

# Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve (  $\star)\,-\,$

**Returns** Generator over Alternative elements

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

# annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

# **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve  $(\star)$  -

### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

# Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables(parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

#### See also:

```
Word.findspans()
```

#### generate id(cls)

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class. set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

# **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

# $\verb"resolveword"\,(id)$

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
\label{text} \begin{tabular}{ll} \textbf{text} (cls='current', retain to ken is at ion=False, previous delimiter=", strict=False, correction handling=1, normalize\_spaces=False) \\ \end{tabular}
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.SemanticRolesLayer

```
class pynlpl.formats.folia.SemanticRolesLayer(doc, *args, **kwargs)
```

Bases: pynlpl.formats.folia.AbstractAnnotationLayer

Syntax Layer: Annotation layer for SemanticRole span annotation elements

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
	Continued on next page

Table 69 – continued from previous page

	ued from previous page
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are multiple)
annotations(Class[, set])	tiple).  Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append() Returns this word in context, {size} words to the left,
<pre>context(size[, placeholder, scope])</pre>	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the specified words or morphemes.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
	ment into an instance of the Class.

Table 69 – continued	from previous	page

	ed from previous page
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
-	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 29
AUTH = True
AUTO_GENERATE_ID = False
```

OCCURRENCES = 0

```
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = False
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = True
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'semroles'
Method Details
init (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
        Raises ValueError
addidsuffix (idsuffix, recursive=True)
    Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
    call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
    Makes sure this element (and all subelements), are properly added to the index.
    Mostly for internal use.
```

### alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*)-

**Returns** Generator over Alternative elements

### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

#### annotation (type, set=None)

Will return a single annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

# annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (\*)-
- set The set you want to retrieve (\*) -

# Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

# context (size, placeholder=None, scope=None)

Returns this word in context,  $\{size\}$  words to the left, the current word, and  $\{size\}$  words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

# **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# correct (\*\*kwargs)

Apply a correction (TODO: documentation to be written still)

# count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

# See also:

```
Word.findspans()
```

# generate\_id(cls)

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

# Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

**hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
\verb|select| (Class, set=None, recursive=True, ignore=True, node=None)|
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# xmlstring (pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

**Return type** str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:

__len__()
Returns the number of child elements under the current element.

__str__()
Alias for text()
```

# pynlpl.formats.folia.TimingLayer

```
class pynlpl.formats.folia.TimingLayer(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractAnnotationLayer
```

Timing layer: Annotation layer for TimeSegment span annotation elements.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
• -	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a
	specific annotation type, and possibly restrained also
	by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
<pre>annotation(type[, set])</pre>	Will return a <b>single</b> annotation (even if there are mul-
	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
	Continued on next page

Table 70 – continued from previous page

	ued from previous page
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
findspan(*words)	Returns the span element which spans over the spec-
	ified words or morphemes.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
_	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
<del></del> -	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
-	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
•	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
<del></del>	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
5 (c	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
1 (	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
	Select child elements of the specified class.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class

Table 70 – continued from previous page

setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the spec-	
	ified class)	
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
<pre>xml([attribs, elements, skipchildren])</pre>	See AbstractElement.xml()	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text ()	

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 23

AUTH = True

AUTO_GENERATE_ID = False

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 10, 11)

PHONCONTAINER = False

PRIMARYELEMENT = False

PRINTABLE = False

REQUIRED_ATTRIBS = None

REQUIRED_DATA = None

SETONLY = True

SPEAKABLE = False

SUBSET = None
```

```
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'timing'
```

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

## Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

### **Parameters**

- Class The Class you want to retrieve (\*) -
- set The set you want to retrieve (\*)-

**Returns** Generator over Alternative elements

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
annotation (type, set=None)
```

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

```
annotations (Class, set=None)
```

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve  $(\star)$  -

### Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## **Returns** int

## deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### findspan (\*words)

Returns the span element which spans over the specified words or morphemes.

#### See also:

```
Word.findspans()
```

## generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

## Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling — Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

**hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

## Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

Class (\*) – The class to select; any python class subclassed off 'AbstractElement', may
also be a tuple of multiple classes. Set to True to constrain to the same class as that of
the current instance, set to None to not constrain at all

• **scope** (\*) – A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

## originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

## See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

#### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use <code>AbstractElement.append()</code> if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

## setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

## **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

#### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

Returns The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

• **cls** (*str*) – The class of the text content to obtain, defaults to current.

• correctionhandling – Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

#### Returns bool

```
toktext (cls='current')
```

Alias for text() with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
    ...
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

Some span annotation elements take *span roles*, depending on their type:

CoreferenceLink	Coreference link.
DependencyDependent	Span role element that marks the dependent in a depen-
	dency relation.
	0 11 1

Continued on next page

Table 71 – continued from previous
------------------------------------

Headspan	The headspan role is used to mark the head of a span
	annotation.

## pynlpl.formats.folia.CoreferenceLink

class pynlpl.formats.folia.CoreferenceLink(doc, \*args, \*\*kwargs)

Bases: pynlpl.formats.folia.AbstractSpanRole

Coreference link. Used in CoreferenceChain

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	initialize seri.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addable(parent), set, raiseexceptions;	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
addid at it in (lasalint, recaisive))	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
added_mach([norecanse])	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
anceses (Classes)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
and de de l'a((elass))	fectively back-tracing its path to the root element.
- annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
(-) F-L,J)	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
, 1 3/	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
	Continued on next page

Table 72 –	continued	from	previous	page

	ed from previous page
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
	tation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
maspiron([eis, suret, correctionnumening])	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
	Is this element part of a correction? If it is, it returns
incorrection()	
	the Correction element (evaluating to True), other-
, (' 1 1'11 \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
parsonni (nout, dot, nivalgo)	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
phon([els, previousaemmer, strict,])	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
phoneomeene ([cis, correctionnal dinig])	
10	this element (of the specified class).  This method will be called after an element is added
postappend()	
([0]	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
setspan(*args)	Sets the span of the span element anew, erases all
occopan (ags)	data inside.
cottout(taxt[clc])	
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
	Continued on next page

Continued on next page

Table 72 – continued from previous page

14.5.0 / = 00.11.1	idea ii eiii pieriede page	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
wrefs([index, recurse])	Returns a list of word references, these can be Words	
	but also Morphemes or Phonemes.	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text()	

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 28
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Coreference Link'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 2, 4, 5)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
```

#### XMLTAG = 'coreferencelink'

## **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

## annotation (type, set=None)

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

```
annotations (Class, set=None)
```

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (  $\star$  ) -
- set The set you want to retrieve (\*)-

#### **Yields** elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

## feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

## **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

## **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

### **hastext** (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you

want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

```
phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()
```

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use <code>AbstractElement.append()</code> if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

## ${\tt resolveword}\,(id)$

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.

- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

## setspan(\*args)

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

## speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

Returns The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

## textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
\textbf{xml} \; (attribs = None, \, elements = None, \, skipchildren = False)
```

```
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
iter ()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.DependencyDependent

```
class pynlpl.formats.folia.DependencyDependent (doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanRole
```

Span role element that marks the dependent in a dependency relation. Used in Dependency.

Headspan in turn is used to mark the head of a dependency relation.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	mindile soil.
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
addable(parentl, set, raiscenceptions])	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
addid3dilia(dsdilla[, lectisive])	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
addeoindex([noicedise])	properly added to the index
ancestor(*Classes)	Find the most immediate ancestor of the specified
uncester (Chastes)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
anecocoro([class])	fectively back-tracing its path to the root element.
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-
annotation(type[, set])	tiple).
annotations(Class[, set])	Obtain annotations.
append(child, *args, **kwargs)	See AbstractElement.append()
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
context(size[, piaceholder, scope])	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
copyentraren([newdoc, idsumx])	element.
correct(**kwargs)	Apply a correction (TODO: documentation to be
Correct(**kwargs)	written still)
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
Count (Class], set, recursive, ignore, node])	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
iinacoirectionnanaiing(cis)	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	internal method to find replaceable elements.
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
getindex(child[, recursive, ignore])	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
getmetadata([KCy])	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-
nasamiocactom(Ciass[, SEL])	
hasphon([cls, strict, correctionhandling])	tation exists, and if so, how many.  Does this element have phonetic content (of the spec-
masphon([cis, suici, correctionnanding])	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
THEOTTECCTOH	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	wise it feturis from
THOUSE CHICA, CHICA, args, Kwargs)	Continued on next page
	Continued on next page

Table 73 –	continued	from	previous	page
		•	p. 0 0 0. 0	P~9-

	ed from previous page
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
<pre>next([Class, scope, reverse])</pre>	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
±	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
1 - (L)	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
,]/	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
repraes (cima, args, avangs)	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
beeparenes()	scop.
setspan(*args)	Sets the span of the span element anew, erases all
seespan (mgs)	data inside.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
speecii_speakei()	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
speecii_sic()	associated with the element.
atriattoyt([als])	Alias for text () with strict=True
stricttext([cls])	
text([cls, retaintokenisation,])	<del>-</del>
	ified class)
text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])	ified class)  Get the text content explicitly associated with this
textcontent([cls, correctionhandling])	ified class)  Get the text content explicitly associated with this element (of the specified class).
<pre>textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	ified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.
textcontent([cls, correctionhandling])	ified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with
<pre>textcontent([cls, correctionhandling])  textvalidation([warnonly])  toktext([cls])</pre>	Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True
<pre>textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	ified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with

Continued on next page

Table 73 – continued from previous page

	1
wrefs([index, recurse])	Returns a list of word references, these can be Words
	but also Morphemes or Phonemes.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Dependent'
OCCURRENCES = 1
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 2, 4, 5)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'dep'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
```

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

## classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

## annotation (type, set=None)

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

## annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

#### **Parameters**

- Class The Class you want to retrieve (  $\star)\,-\,$
- set The set you want to retrieve (\*)-

## Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

## Returns str or list

## ${\tt findcorrection handling}\,(\mathit{cls})$

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
\mathtt{generate\_id}\left(\mathit{cls}\right)
```

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

## Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## **Parameters**

```
    node - XML Element (*) -
    doc - Document (*) -
```

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.

- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

## Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

## previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

## setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

Parameters \*args - Instances of Word, Morpheme or Phoneme

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

## speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• **cls** (str) – The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (int or None) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
See AbstractElement.xml ()
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

## Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
    ...
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.Headspan

```
class pynlpl.formats.folia.Headspan(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractSpanRole
```

The headspan role is used to mark the head of a span annotation.

It can be used in various contexts, for instance to mark the head of a <code>Dependency</code>. It is allowed by most span annotations.

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index
	0 1' 1

Continued on next page

Table 74	l – continued	from	prev	ious page

Table 74 – continued from previous page				
ancestor(*Classes)	Find the most immediate ancestor of the specified			
	type, multiple classes may be specified.			
ancestors([Class])	Generator yielding all ancestors of this element, ef-			
	fectively back-tracing its path to the root element.			
annotation(type[, set])	Will return a <b>single</b> annotation (even if there are mul-			
(AL-F) 3)	tiple).			
annotations(Class[, set])	Obtain annotations.			
append(child, *args, **kwargs)	See AbstractElement.append()			
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,			
context(size[, placeholder, scope])	the current word, and {size} words to the right			
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.			
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this			
copychitaren([newdoc, ldsumx])	element.			
correct(**kwargs)	Apply a correction (TODO: documentation to be			
, , , ,	written still)			
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-			
( 1, , , , , , , , , , , , , , , , , , ,	stead of returning the elements, it merely counts			
	them.			
deepvalidation()	Perform deep validation of this element.			
description()	Obtain the description associated with the element.			
feat(subset)	Obtain the feature class value of the specific subset.			
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass			
i inacorrectionnanaring (cis)	by looking in the underlying corrections where it is			
	reused			
findreplaceables(parent[, set])	Internal method to find replaceable elements.			
generate_id(cls)	internal method to find replaceable elements.			
	Cat the index at which are also and a source provide			
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!			
getmetadata([key])	Get the metadata that applies to this element, auto-			
	matically inherited from parent elements			
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.			
hasannotation(Class[, set])	Returns an integer indicating whether such as anno-			
( 1, 3,	tation exists, and if so, how many.			
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-			
61)	ified class)			
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)			
incorrection()	Is this element part of a correction? If it is, it returns			
	the Correction element (evaluating to True), other-			
	wise it returns None			
insert(index, child, *args, **kwargs)				
items([founditems])	Returns a depth-first flat list of all items below this			
± 6 cmo([roundioms])	element (not limited to AbstractElement)			
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a			
Journal Jacobs, recurse, ignoralist])				
loft contout(size[ placeholder come))	Python dictionary suitable for serialisation to JSON.			
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.			
next([Class, scope, reverse])	Returns the next element, if it is of the specified type			
	and if it does not cross the boundary of the defined			
7 ( 1 1)	scope.			
originaltext([cls])	Alias for retrieving the original uncorrect text.			
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-			
	ment into an instance of the Class.			
	Continued on next page			

Table 74 – continued from previous page				
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this			
	element (of the specified class)			
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with			
-	this element (of the specified class).			
postappend()	This method will be called after an element is added			
	to another and does some checks.			
previous([Class, scope])	Returns the previous element, if it is of the specified			
- · · · · · · · · · · · · · · · · · · ·	type and if it does not cross the boundary of the de-			
	fined scope.			
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an			
	XML element (lxml.etree) rather than a string)			
remove(child)	Removes the child element			
replace(child, *args, **kwargs)	Appends a child element like append(), but re-			
	places any existing child element of the same type			
	and set.			
resolveword(id)				
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.			
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.			
setdoc(newdoc)	Set a different document.			
setdocument(doc)	Associate a document with this element.			
setparents()	Correct all parent relations for elements within the			
	scop.			
setspan(*args)	Sets the span of the span element anew, erases all			
	data inside.			
settext(text[, cls])	Set the text for this element.			
speech_speaker()	Retrieves the speaker of the audio or video file asso-			
	ciated with the element.			
speech_src()	Retrieves the URL/filename of the audio or video file			
	associated with the element.			
stricttext([cls])	Alias for text() with strict=True			
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-			
	ified class)			
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this			
	element (of the specified class).			
textvalidation([warnonly])	Run text validation on this element.			
toktext([cls])	Alias for text() with			
	retaintokenisation=True			
updatetext()	Recompute textual value based on the text content of			
	the children.			
wrefs([index, recurse])	Returns a list of word references, these can be Words			
	but also Morphemes or Phonemes.			
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()			
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to			
	XML.			
iter()	Iterate over all children of this element.			
len()	Returns the number of child elements under the cur-			
	rent element.			
str()	Alias for text()			

## **Class Attributes**

ACCEPTED\_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats

```
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Head'
OCCURRENCES = 1
OCCURRENCES PER SET = 0
OPTIONAL\_ATTRIBS = (0, 2, 4, 5)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'hd'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
```

Raises ValueError

#### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by *copy* ()

#### addtoindex (norecurse=None)

Makes sure this element (and all subelements), are properly added to the index

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

### annotation (type, set=None)

Will return a **single** annotation (even if there are multiple). Raises a NoSuchAnnotation exception if none was found

#### annotations (Class, set=None)

Obtain annotations. Very similar to select () but raises an error if the annotation was not found.

# **Parameters**

- Class The Class you want to retrieve(\*)-
- set The set you want to retrieve (\*)-

# Yields elements

Raises NoSuchAnnotation if the specified annotation does not exist.

```
append (child, *args, **kwargs)
```

```
See AbstractElement.append()
```

#### context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

# **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element. If idsuffix is a string, if set to True, a random idsuffix will be generated including a random 32-bit hash

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

# count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### hasannotation(Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many. See annotations () for a description of the parameters.

# hasphon (cls='current', strict=True, correctionhandling=1)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# Parameters

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

#### originaltext(cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

## classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

#### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

• alternative (bool) – If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element

• be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

# Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by <code>copy()</code>

```
setspan(*args)
```

Sets the span of the span element anew, erases all data inside.

```
Parameters *args - Instances of Word, Morpheme or Phoneme
```

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

#### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text() with retaintokenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
wrefs (index=None, recurse=True)
```

Returns a list of word references, these can be Words but also Morphemes or Phonemes.

**Parameters index** (*int or None*) – If set to an integer, will retrieve and return the n'th element (starting at 0) instead of returning the list of all

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

### Return type str

```
iter ()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

```
__len__()
Returns the number of child elements under the current element.
__str__()
Alias for text()
```

# 4.2 Editing FoLiA

# 4.2.1 Creating a new document

Creating a new FoliA document, rather than loading an existing one from file, is done by explicitly providing the ID for the new document in the Document constructor:

```
doc = folia.Document(id='example')
```

# 4.2.2 Declarations

Whenever you add a new **type** of annotation, or a different set, to a FoLiA document, you have to first declare it. This is done using the *Document.declare()* method. It takes as arguments the annotation type, the set, and you can optionally pass keyword arguments to annotator= and annotatortype= to set defaults.

An example for Part-of-Speech annotation:

```
doc.declare(folia.PosAnnotation, 'http://somewhere/brown-tag-set')
```

An example with a default annotator:

Any additional sets for Part-of-Speech would have to be explicitly declared as well. To check if a particular annotation type and set is declared, use the <code>Document.declared()</code> method.

# 4.2.3 Adding structure

Assuming we begin with an empty document, we should first add a Text element. Then we can add paragraphs, sentences, or other structural elements. The <code>AbstractElement.add()</code> method adds new children to an element:

```
text = doc.add(folia.Text)
paragraph = text.add(folia.Paragraph)
sentence = paragraph.add(folia.Sentence)
sentence.add(folia.Word, 'This')
sentence.add(folia.Word, 'is')
sentence.add(folia.Word, 'a')
sentence.add(folia.Word, 'test')
sentence.add(folia.Word, '.')
```

**Note:** The <code>AbstractElement.add()</code> method is actually a wrapper around <code>AbstractElement.append()</code>, which takes the exact same arguments. It performs extra checks and works for both span annotation as well as token annotation. Using <code>append()</code> will be faster though.

4.2. Editing FoLiA 797

# 4.2.4 Adding annotations

Adding annotations, or any elements for that matter, is done using the <code>AbstractElement.add()</code> method on the intended parent element. We assume that the annotations we add have already been properly declared, otherwise an exception will be raised as soon as <code>add()</code> is called. Let's build on the previous example:

```
#First we grab the fourth word, 'test', from the sentence
word = sentence.words(3)

#Add Part-of-Speech tag
word.add(folia.PosAnnotation, set='brown-tagset',cls='n')

#Add lemma
lemma.add(folia.LemmaAnnotation, cls='test')
```

Note that in the above examples, the add () method takes a class as first argument, and subsequently takes keyword arguments that will be passed to the classes' constructor.

A second way of using AbstractElement.add() is by simply passing a fully instantiated child element, thus constructing it prior to adding. The following is equivalent to the above example, as the previous method is merely a shortcut for convenience:

```
#First we grab the fourth word, 'test', from the sentence
word = sentence.words(3)

#Add Part-of-Speech tag
word.add( folia.PosAnnotation(doc, set='brown-tagset',cls='n') )

#Add lemma
lemma.add( folia.LemmaAnnotation(doc , cls='test') )
```

The AbstractElement.add() method always returns that which was added, allowing it to be chained.

In the above example we first explicitly instantiate a *PosAnnotation* and a *LemmaAnnotation*. Instantiation of any FoLiA element (always Python class subclassed off *AbstractElement*) follows the following pattern:

```
Class(document, *children, **kwargs)
```

Note: See AbstractElement.\_\_init\_\_ () for all details on construction

Note that the document has to be passed explicitly as first argument to the constructor.

The common attributes are set using equally named keyword arguments:

- id=
- cls=
- set=
- annotator=
- annotatortype=
- confidence=
- src=
- speaker=

- begintime=
- endtime=

Not all attributes are allowed for all elements, and certain attributes are required for certain elements. ValueError exceptions will be raised when these constraints are not met.

Instead of setting id. you can also set the keyword argument <code>generate\_id\_in</code> and pass it another element, an ID will be automatically generated, based on the ID of the element passed. When you use the first method of adding elements, instantiation with <code>generate\_id\_in</code> will take place automatically behind the scenes when applicable and when <code>id</code> is not explicitly set.

Any extra non-keyword arguments should be FoLiA elements and will be appended as the contents of the element, i.e. the children or subelements. Instead of using non-keyword arguments, you can also use the keyword argument content and pass a list. This is a shortcut made merely for convenience, as Python obliges all non-keyword arguments to come before the keyword-arguments, which if often aesthetically unpleasing for our purposes. Example of this use case will be shown in the next section.

# 4.2.5 Adding span annotation

Adding span annotation is easy with the FoLiA library. As you know, span annotation uses a stand-off annotation embedded in annotation layers. These layers are in turn embedded in structural elements such as sentences. However, the <code>AbstractElement.add()</code> method abstracts over this. Consider the following example of a named entity:

To make references to the words, we simply pass the word instances and use the document's index to obtain them. Note also that passing a list using the keyword argument contents is wholly equivalent to passing the non-keyword arguments separately:

```
word.add(folia.Entity, cls="per", contents=[word,word2])
```

In the next example we do things more explicitly. We first create a sentence and then add a syntax parse, consisting of nested elements:

```
doc.declare(folia.SyntaxLayer, 'some-syntax-set')

sentence = text.add(folia.Sentence)
sentence.add(folia.Word, 'The',id='example.s.1.w.1')
sentence.add(folia.Word, 'boy',id='example.s.1.w.2')
sentence.add(folia.Word, 'pets',id='example.s.1.w.3')
sentence.add(folia.Word, 'the',id='example.s.1.w.4')
sentence.add(folia.Word, 'cat',id='example.s.1.w.5')
sentence.add(folia.Word, '.', id='example.s.1.w.6')

#Adding Syntax Layer
```

(continues on next page)

4.2. Editing FoLiA 799

(continued from previous page)

```
layer = sentence.add(folia.SyntaxLayer)
#Adding Syntactic Units
layer.add(
    folia.SyntacticUnit(self.doc, cls='s', contents=[
        folia.SyntacticUnit(self.doc, cls='np', contents=[
            folia.SyntacticUnit(self.doc, self.doc['example.s.1.w.1'], cls='det'),
            folia.SyntacticUnit(self.doc, self.doc['example.s.1.w.2'], cls='n'),
        ]),
        folia.SyntacticUnit(self.doc, cls='vp', contents=[
            folia.SyntacticUnit(self.doc, self.doc['example.s.1.w.3'], cls='v')
                folia.SyntacticUnit(self.doc, cls='np', contents=[
                     folia.SyntacticUnit(self.doc, self.doc['example.s.1.w.4'], cls=
\rightarrow 'det'),
                     folia.SyntacticUnit(self.doc, self.doc['example.s.1.w.5'], cls='n
\hookrightarrow '),
                ]),
            ]),
        folia.SyntacticUnit(self.doc, self.doc['example.s.1.w.6'], cls='fin')
    ])
```

**Note:** The lower-level <code>AbstractElement.append()</code> method would have had the same effect in the above syntax tree sample.

# 4.2.6 Deleting annotations

Any element can be deleted by calling the <code>AbstractElement.remove()</code> method on its parent. Suppose we want to delete word:

```
word.parent.remove(word)
```

# 4.2.7 Copying annotations

A deep copy can be made of any element by calling its AbstractElement.copy() method:

```
word2 = word.copy()
```

The copy will be without parent and document. If you intend to associate a copy with a new document, then copy as follows instead:

```
word2 = word.copy(newdoc)
```

If you intend to attach the copy somewhere in the same document, you may want to add a suffix for any identifiers in its scope, since duplicate identifiers are not allowed and would raise an exception. This can be specified as the second argument:

```
word2 = word.copy(doc, ".copy")
```

# 4.3 Searching in a FoLiA document

If you have loaded a FoLiA document into memory, you may want to search for a particular annotations. You can of course loop over all structural and annotation elements using <code>AbstractElement.select()</code>, <code>AllowTokenAnnotation.annotation()</code> and <code>AllowTokenAnnotation.annotations()</code>. Additionally, <code>Word.findspans()</code> and <code>AbstractAnnotationLayer.findspan()</code> are useful methods of finding span annotations covering particular words, whereas <code>AbstractSpanAnnotation.wrefs()</code> does the reverse and finds the words for a given span annotation element. In addition to these main methods of navigation and selection, there is higher-level function available for searching, this uses the <code>FoLiA Query Language(FQL)</code> or the <code>Corpus Query Language(CQL)</code>.

These two languages are part of separate libraries that need to be imported:

```
from pynlpl.formats import fql, cql
```

# 4.3.1 Corpus Query Language (CQL)

CQL is the easier-language of the two and most suitable for corpus searching. It is, however, less flexible than FQL, which is designed specifically for FoLiA and can not just query, but also manipulate FoLiA documents in great detail.

CQL was developed for the IMS Corpus Workbench, at Stuttgart University, and is implemented in Sketch Engine, who provide good CQL documentation.

CQL has to be converted to FQL first, which is then executed on the given document. This is a simple example querying for the word "house":

```
doc = folia.Document(file="/path/to/some/document.folia.xml")
query = fql.Query(cql.cql2fql('"house"'))
for word in query(doc):
    print(word) #these will be folia.Word instances (all matching house)
```

Multiple words can be queried:

```
query = fql.Query(cql.cql2fql('"the" "big" "house"'))
for word1,word2,word3 in query(doc):
    print(word1, word2,word3)
```

Queries may contain wildcard expressions to match multiple text patterns. Gaps can be specified using []. The following will match any three word combination starting with the and ending with something that starts with house. It will thus match things like "the big house" or "the small household":

```
query = fql.Query(cql.cql2fql('"the" [] "house.*"'))
for word1,word3 in query(doc):
    ...
```

We can make the gap optional with a question mark, it can be lenghtened with + or \*, like regular expressions:

```
query = fql.Query(cql.cql2fql('"the" []? "house.*"'))
for match in query(doc):
    print("We matched ", len(match), " words")
```

Querying is not limited to text, but all of FoLiA's annotations can be used. To force our gap consist of one or more adjectives, we do:

```
query = fql.Query(cql.cql2fql('"the" [ pos = "a" ]+ "house.*"'))
for match in query(doc):
    ...
```

The original CQL attribute here is tag rather than pos, this can be used too. In addition, all FoLiA element types can be used! Just use their FoLiA tagname.

Consult the CQL documentation for more. Do note that CQL is very word/token centered, for searching other types of elements, use FQL instead.

# 4.3.2 FoLiA Query Language (FQL)

FQL is documented here, a full overview is beyond the scope of this documentation. We will just introduce some basic selection queries so you can develop an initial impression of the language's abilities.

All FQL processing is done via the following class, as already seen in the previous section:

Query

This class represents an FQL query.

# pynlpl.formats.fql.Query

```
class pynlpl.formats.fql.Query(q, context=<pynlpl.formats.fql.Context object>)
    Bases: object
```

This class represents an FQL query.

Selecting a word with a particular text is done as follows, doc is an instance of pynlpl.formats.folia.

Document:

```
query = fql.Query('SELECT w WHERE text = "house"')
for word in query(doc):
    print(word) #this will be an instance of folia.Word
```

Regular expression matching can be done using the MATCHES operator:

```
query = fql.Query('SELECT w WHERE text MATCHES "^house.*$"')
for word in query(doc):
    print(word)
```

The classes of other annotation types can be easily queried as follows:

```
query = fql.Query('SELECT w WHERE :pos = "v"' AND :lemma = "be"')
for word in query(doc):
    print(word)
```

You can constrain your queries to a particular target selection using the FOR keyword:

```
query = fql.Query('SELECT w WHERE text MATCHES "^house.*$" FOR s WHERE text_
→CONTAINS "sell"')
for word in query(doc):
    print(word)
```

This construction also allows you to select the actual annotations. To select all people (a named entity) for words that are not John:

**FOR** statement may be chained, and Explicit IDs can be passed using the ID keyword:

Sets are specified using the **OF** keyword, it can be omitted if there is only one for the annotation type, but will be required otherwise:

```
query = fql.Query('SELECT su OF "http://some/syntax/set" WHERE class = "np"')
for su in query(doc):
    print(su) #this will be an instance of folia.SyntacticUnit
```

We have just covered just the **SELECT** keyword, FQL has other keywords for manipulating documents, such as **EDIT**, **ADD**, **APPEND** and **PREPEND**.

**Note:** Consult the FQL documentation at https://github.com/proycon/foliadocserve/blob/master/README.rst for further documentation on the language.

# **Method Summary**

```
__init__(q[, context]) Initialize self.

parse(q[, i])
```

## **Method Details**

Selecting a word with a particular text is done as follows:

```
query = fql.Query('SELECT w WHERE text = "house"')
for word in query(doc):
    print(word) #this will be an instance of folia.Word
```

Regular expression matching can be done using the MATCHES operator:

```
query = fql.Query('SELECT w WHERE text MATCHES "^house.*$"')
for word in query(doc):
    print(word)
```

The classes of other annotation types can be easily queried as follows:

```
query = fql.Query('SELECT w WHERE :pos = "v"' AND :lemma = "be"')
for word in query(doc):
    print(word)
```

You can constrain your queries to a particular target selection using the FOR keyword:

```
query = fql.Query('SELECT w WHERE text MATCHES "^house.*$" FOR s WHERE text CONTAINS
    →"sell"')
for word in query(doc):
    print(word)
```

This construction also allows you to select the actual annotations. To select all people (a named entity) for words that are not John:

```
query = fql.Query('SELECT entity WHERE class = "person" FOR w WHERE text != "John"')
for entity in query(doc):
    print(entity) #this will be an instance of folia.Entity
```

**FOR** statement may be chained, and Explicit IDs can be passed using the ID keyword:

```
query = fql.Query('SELECT entity WHERE class = "person" FOR w WHERE text != "John"_
→FOR div ID "section.21"')
for entity in query(doc):
    print(entity)
```

Sets are specified using the **OF** keyword, it can be omitted if there is only one for the annotation type, but will be required otherwise:

```
query = fql.Query('SELECT su OF "http://some/syntax/set" WHERE class = "np"')
for su in query(doc):
    print(su) #this will be an instance of folia.SyntacticUnit
```

We have just covered the **SELECT** keyword, FQL has other keywords for manipulating documents, such as **EDIT**, **ADD**, **APPEND** and **PREPEND**.

**Note:** Consult the FQL documentation at https://github.com/proycon/foliadocserve/blob/master/README.rst for further documentation on the language.

# 4.3.3 Streaming Reader

Throughout this tutorial you have seen the *Document* class as a means of reading FoLiA documents. This class always loads the entire document in memory, which can be a considerable resource demand. The following class provides an alternative to loading FoLiA documents:

Reader Streaming FoLiA reader.

### pynlpl.formats.folia.Reader

```
class pynlpl.formats.folia.Reader (filename, target, *args, **kwargs)
    Bases: object
    Streaming FoLiA reader.
```

The reader allows you to read a FoLiA Document without holding the whole tree structure in memory. The document will be read and the elements you seek returned as they are found. If you are querying a corpus of large FoLiA documents for a specific structure, then it is strongly recommend to use the Reader rather than the standard Document!

# **Method Summary**

init(filename, target, *args, **kwargs)	Read a FoLiA document in a streaming fashion.
findwords(*args, **kwargs)	
initdoc()	

#### **Method Details**

```
__init__ (filename, target, *args, **kwargs)
```

Read a FoLiA document in a streaming fashion. You select a specific target element and all occurrences of this element, including all contents (so all elements within), will be returned.

# **Parameters**

- **filename** (\*) The filename of the document to read
- target (\*) The FoLiA element(s) you want to read (with everything contained in its scope). Passed as a class. For example: folia.Sentence, or a tuple of multiple element classes. Can also be set to None to return all elements, but that would load the full tree structure into memory.

```
__init__ (filename, target, *args, **kwargs)
```

Read a FoLiA document in a streaming fashion. You select a specific target element and all occurrences of this element, including all contents (so all elements within), will be returned.

#### **Parameters**

- **filename** (\*) The filename of the document to read
- target (\*) The FoLiA element(s) you want to read (with everything contained in its scope). Passed as a class. For example: folia.Sentence, or a tuple of multiple element classes. Can also be set to None to return all elements, but that would load the full tree structure into memory.

```
findwords (*args, **kwargs)
initdoc()
```

It does not load the entire document in memory but merely returns the elements you are interested in. This results in far less memory usage and also provides a speed-up.

A reader is constructed as follows, the second argument is the class of the element you want:

```
reader = folia.Reader("my.folia.xml", folia.Word)
for word in reader:
    print(word.id)
```

# 4.4 Higher-Order Annotations

# 4.4.1 Text Markup

FoLiA has a number of text markup elements, these appear within the <code>TextContent</code> (t) element, iterating over the element of a <code>TextContent</code> element will first and foremost produce strings, but also uncover these markup elements when present. The following markup types exists:

TextMarkupGap	Markup element to mark gaps in text content
	(TextContent)
TextMarkupString	Markup element to mark arbitrary substrings in text con-
	<pre>tent (TextContent)</pre>
TextMarkupStyle	Markup element to style text content (TextContent),
	e.g.
TextMarkupCorrection	Markup element to mark corrections in text content
	(TextContent).
TextMarkupError	Markup element to mark gaps in text content
	(TextContent)

# pynlpl.formats.folia.TextMarkupGap

class pynlpl.formats.folia.TextMarkupGap(doc, \*args, \*\*kwargs)

Bases: pynlpl.formats.folia.AbstractTextMarkup

Markup element to mark gaps in text content (TextContent)

Only consider this element for gaps in spans of untokenised text. The use of structural element Gap is preferred.

# **Method Summary**

init(doc, *args, **kwargs)	See AbstractElementinit(), text is
	passed as a string in *args.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
	Continued on next page
	a a series a series de la granda

Table 80 – continued from previous page	Table 80 –	continued	from	previous	page
-----------------------------------------	------------	-----------	------	----------	------

	ed from previous page
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
•	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
, (L)	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
V	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
1/	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	See AbstractElement.json()
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
1 / 1/	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
1 1/	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
5/F	
	· ·
remove(child)	XML element (lxml.etree) rather than a string) Removes the child element
remove(child) replace(child, *args, **kwargs)	XML element (lxml.etree) rather than a string) Removes the child element
remove(child) replace(child, *args, **kwargs)	XML element (lxml.etree) rather than a string) Removes the child element Appends a child element like append(), but re-
	XML element (lxml.etree) rather than a string) Removes the child element Appends a child element like append(), but replaces any existing child element of the same type
replace(child, *args, **kwargs)	XML element (lxml.etree) rather than a string) Removes the child element Appends a child element like append(), but re-
replace(child, *args, **kwargs)  resolve()	XML element (lxml.etree) rather than a string) Removes the child element Appends a child element like append(), but replaces any existing child element of the same type
replace(child, *args, **kwargs)  resolve()  resolveword(id)	XML element (lxml.etree) rather than a string) Removes the child element Appends a child element like append(), but replaces any existing child element of the same type and set.
replace(child, *args, **kwargs)  resolve()	XML element (lxml.etree) rather than a string) Removes the child element Appends a child element like append(), but replaces any existing child element of the same type

Table 80 – continued from previous page

setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
setparents()	_	
	scop.	
settext(text)	Sets the text content of the markup element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
· · ·	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
3 <b>~1</b> • • • •	XML.	
iter()	Iterate over all children of this element.	
len_()	Returns the number of child elements under the cur-	
·	rent element.	
str()	Alias for text()	

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractTextMarkup'>, <class 'pynlpl.formats.folia.AbstractTex
```

```
TEXTCONTAINER = True
TEXTDELIMITER = ''
XLINK = True
XMLTAG = 't-gap'
```

#### **Method Details**

```
__init__ (doc, *args, **kwargs)
See AbstractElement.__init__ (), text is passed as a string in *args.
__init__ (doc, *args, **kwargs)
See AbstractElement.__init__ (), text is passed as a string in *args.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

# Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

# Returns int

#### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

```
See AbstractElement.json()
```

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
node - XML Element (*) -doc - Document (*) -
```

Returns An instance of the current Class.

**phon** (*cls='current'*, *previousdelimiter=", strict=False*, *correctionhandling=1*)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolve()
```

resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

#### Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

#### settext (text)

Sets the text content of the markup element.

```
Parameters text (str) -
```

#### speech speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
```

Alias for text() with strict=True

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

## Parameters

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

#### Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

```
See AbstractElement.xml()
```

# $\mathbf{xmlstring} \ (pretty\_print = False)$

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.TextMarkupString

class pynlpl.formats.folia.TextMarkupString(doc, \*args, \*\*kwargs)
 Bases: pynlpl.formats.folia.AbstractTextMarkup

Markup element to mark arbitrary substrings in text content (TextContent)

# **Method Summary**

( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )	0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
init(doc, *args, **kwargs)	See AbstractElementinit(), text is
	passed as a string in *args.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
	Continued on next page

Table 81 – continu	ued from previous page
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	See AbstractElement.json()
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
-	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
-	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolve()	
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text)	Sets the text content of the markup element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
	0 11 1

Continued on next page

Table 81 – continued from previous page

Table 61 Continues	a nom providuo pago
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractTextMarkup'>, <class 'pynlpl.folia.AbstractTextMarkup'>, <class 'pynlpl.fo
ANNOTATIONTYPE = 32
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = True
TEXTDELIMITER = ''
XLINK = True
XMLTAG = 't-str'
Method Details
 __init__(doc, *args, **kwargs)
               See AbstractElement.__init__ (), text is passed as a string in *args.
__init__ (doc, *args, **kwargs)
               See AbstractElement.__init__(), text is passed as a string in *args.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
```

add (child, \*args, \*\*kwargs)

#### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

#### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

### copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

#### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
See AbstractElement.json()
```

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) doc Document (\*) -
- Returns An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

#### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolve()

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*) Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

## setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text)
```

Sets the text content of the markup element.

```
Parameters text (str) -
```

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

Returns The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

# **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.TextMarkupStyle

```
class pynlpl.formats.folia.TextMarkupStyle(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractTextMarkup
```

Markup element to style text content (TextContent), e.g. make text bold, italics, underlined, coloured, etc..

### **Method Summary**

```
__init__(doc, *args, **kwargs) See AbstractElement.__init__(), text is passed as a string in *args.
```

Continued on next page

Table 82 – continued from previous page

	ed from previous page
<pre>accepts(Class[, raiseexceptions, parentinstance])</pre>	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	See AbstractElement.json()
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined
original+ov+([cls])	Scope.  Alias for retriaving the original uncorrect text
<pre>originaltext([cls]) parsexml(node, doc, **kwargs)</pre>	Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML ele-
parseaux (node, doc, kwaigs)	ment into an instance of the Class.
	Continued on next page

Continued on next page

Table 82 – continu	ied from previous page
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
phoneometre ([cis, correctionnanding])	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolve()	
resolveword(id)	
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.
<pre>select(Class[, set, recursive, ignore, node])</pre>	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text)	Sets the text content of the markup element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
<pre>xml([attribs, elements, skipchildren])</pre>	See AbstractElement.xml()
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.

# **Class Attributes**

\_\_str\_\_()

ACCEPTED\_DATA = (<class 'pynlpl.formats.folia.AbstractTextMarkup'>, <class 'pynlpl.formats.folia.AbstractTex

rent element.

Alias for text ()

Returns the number of child elements under the cur-

AUTH = True

```
AUTO GENERATE ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = True
TEXTDELIMITER = ''
XLINK = True
XMLTAG = 't-style'
Method Details
__init__ (doc, *args, **kwargs)
    See AbstractElement.__init__ (), text is passed as a string in *args.
__init__ (doc, *args, **kwargs)
    See AbstractElement.__init__ (), text is passed as a string in *args.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) – Raise an exception if the element can't be added?
        Returns bool
        Raises ValueError
addidsuffix (idsuffix, recursive=True)
    Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
```

call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

## **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

# deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

# description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### **Returns** str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

 $\verb|hastext| (cls='current', strict=True, correction handling=1)|$ 

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling – Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
See AbstractElement.json()
```

```
leftcontext (size, placeholder=None, scope=None)
```

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

```
classmethod parsexml (node, doc, **kwargs)
```

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
node - XML Element (*) -doc - Document (*) -
```

Returns An instance of the current Class.

```
\textbf{phon} \ (cls='current', previous de limiter=", strict=False, correction hand ling=1")
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

• cls (str) – The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolve()

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.

• **node** (\*) – Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

#### settext (text)

Sets the text content of the markup element.

```
Parameters text (str) -
```

#### speech speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

# speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext(cls='current')
```

```
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.

- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### Parameters 2 4 1

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

## textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

#### Returns bool

## toktext (cls='current')

Alias for text () with retaintokenisation=True

## updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
See AbstractElement.xml()
```

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.TextMarkupCorrection

```
class pynlpl.formats.folia.TextMarkupCorrection(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractTextMarkup
```

Markup element to mark corrections in text content (TextContent).

Only consider this element for corrections on untokenised text. The use of Correction is preferred.

# **Method Summary**

init(doc, *args, **kwargs)	See AbstractElementinit(), text is
	passed as a string in *args.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
	Continued on next page

Chapter 4. FoLiA library

T 1 1 00		•		
Table 83 -	CONTINUED	trom	nravinie	nana
Table 00	CONTINUCA	11 0111	picvious	page

	ued from previous page
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
	by default!
<pre>getmetadata([key])</pre>	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	See AbstractElement.json()
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
1 / 1/	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
2 · · · · · · · · · · · · · · · · · · ·	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
· · · · · · · · · · · · · · ·	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
Z (C ) (T ) (T )	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
accomo, argo, mago,	places any existing child element of the same type
	and set.
resolve()	
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
	Select child elements of the specified class.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class

Table 83 – continued from previous page

setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text)	Sets the text content of the markup element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
$t \in xt([cls, retaintokenisation,])$	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for $text()$ with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xml([attribs, elements, skipchildren])	See AbstractElement.xml()
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractTextMarkup'>, <class 'pynlpl.formats.folia.AbstractTex
```

```
TEXTCONTAINER = True

TEXTDELIMITER = ''

XLINK = True

XMLTAG = 't-correction'
```

### **Method Details**

```
__init__ (doc, *args, **kwargs)
    See AbstractElement.__init__ (), text is passed as a string in *args.
__init__ (doc, *args, **kwargs)
    See AbstractElement.__init__ (), text is passed as a string in *args.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

## Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

## addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str* or *bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

## deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

# ${\tt findcorrection handling} \ ({\it cls})$

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

# Returns int

### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

## **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

```
See AbstractElement.json()
```

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
node - XML Element (*) -doc - Document (*) -
```

Returns An instance of the current Class.

phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

## **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolve()
```

resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

#### settext (text)

Sets the text content of the markup element.

```
Parameters text (str) -
```

### speech speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

## speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
```

Alias for text () with strict=True

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

## **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *text* (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

See AbstractElement.xml()

# $\mathbf{xmlstring} \ (pretty\_print = False)$

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
iter ()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.TextMarkupError

class pynlpl.formats.folia.TextMarkupError(doc, \*args, \*\*kwargs)

 $Bases: \verb|pynlpl.formats.folia.AbstractTextMarkup||$ 

Markup element to mark gaps in text content (TextContent)

Only consider this element for gaps in spans of untokenised text. The use of structural element ErrorDetection is preferred.

# **Method Summary**

init(doc, *args, **kwargs)	See AbstractElementinit(), text is
	passed as a string in *args.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	• •
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
, 1 3	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
([,])	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
gooding, recarsive, ignorej)	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
geemeeadaea([hey])	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
gettextuerrmitter([retaintokemsation])	
	Continued on next page

	ed from previous page
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	See AbstractElement.json()
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
_	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
, ,	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolve()	
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text)	Sets the text content of the markup element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
<del>"</del>	ciated with the element.
speech_src()	D
- "	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	
	associated with the element.  Alias for text() with strict=True
<pre>stricttext([cls]) text([cls, retaintokenisation,])</pre>	associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the spec-
text([cls, retaintokenisation,])	associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)
	associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this
text([cls, retaintokenisation,])	associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)

Table 84 – continued from previous page

toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xml([attribs, elements, skipchildren])	See AbstractElement.xml()
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractTextMarkup'>, <class 'pynlpl.folia.AbstractTextMarkup'>, <class 'pynlpl.fo
ANNOTATIONTYPE = 17
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = False
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = True
TEXTDELIMITER = ''
XLINK = True
XMLTAG = 't-error'
Method Details
__init__ (doc, *args, **kwargs)
               See AbstractElement.__init__ (), text is passed as a string in *args.
___init___(doc, *args, **kwargs)
               See AbstractElement.__init__(), text is passed as a string in *args.
```

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by COPY()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

#### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

# count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

## findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# hasphon(cls='current', strict=True, correctionhandling=1)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
See AbstractElement.json()
```

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) doc Document (\*) -
- Returns An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

## remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolve()

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*) Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

## setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

## setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text)
```

Sets the text content of the markup element.

```
Parameters text (str) -
```

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

# **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

```
Alias for text() with retaintokenisation=True
```

```
updatetext()
```

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
See AbstractElement.xml ()
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

\_\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# 4.4.2 Features

Features allow a second-order annotation by adding the ability to assign properties and values to any of the existing annotation elements. They follow the set/class paradigm by adding the notion of a subset and class relative to this subset. The <code>AbstractElement.feat()</code> method provides a shortcut that can be used on any annotation element to obtain the class of the feature, given a subset. To illustrate the concept, take a look at part of speech annotation with some features:

```
pos = word.annotation(folia.PosAnnotation)
if pos.cls = "n":
    if pos.feat('number') == 'plural':
        print("We have a plural noun!")
    elif pos.feat('number') == 'singular':
        print("We have a singular noun!")
```

The AbstractElement.feat() method will return an exception when the feature does not exist. Note that the actual subset and class values are defined by the set and not FoLiA itself! They are therefore fictitious in the above example.

The Python class for features is Feature, in the following example we add a feature:

```
pos.add(folia.Feature, subset="gender", cls="f")
```

Although FoLiA does not define any sets nor subsets. Some annotation types do come with some associated subsets, their use is never mandatory. The advantage is that these associated subsets can be directly used as an XML attribute in the FoLiA document. The FoLiA library provides extra classes, all subclassed off Feature for these:

Feature	Feature elements can be used to associate subsets and subclasses with almost any annotation element
SynsetFeature	Synset feature, to be used within Sense
ActorFeature	Actor feature, to be used within Event
BegindatetimeFeature	Begindatetime feature, to be used within Event
EnddatetimeFeature	Enddatetime feature, to be used within Event

# pynlpl.formats.folia.Feature

```
class pynlpl.formats.folia.Feature(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Feature elements can be used to associate subsets and subclasses with almost any annotation element

# **Method Summary**

init(doc, *args, **kwargs)	Constructor.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
	Continued on next page

Table 86 – continued from previous page

description() feat(subset)	Obtain the description associated with the element.
* /	
	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
<pre>getmetadata([key])</pre>	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
<pre>hastext([cls, strict, correctionhandling])</pre>	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
1()	this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the de- fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	Datums the right contact for an element, as a list
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
t - 1 t (1 )	Associate a document with this element.
setdocument(doc) setparents()	Correct all parent relations for elements within the

Table 86 – continued from previous page

Table 35 Serial as	a nom providuo pago
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xml()	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

SPEAKABLE = False

TEXTCONTAINER = False

SUBSET = None

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Feature'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = None
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
```

```
TEXTDELIMITER = None

XLINK = False

XMLTAG = 'feat'

Method Details

__init__(doc, *args, **kwargs)

Constructor.
```

# **Keyword Arguments**

- **subset** (str) the subset
- cls(str) the class

```
__init__(doc, *args, **kwargs)
```

Constructor.

## **Keyword Arguments**

- **subset** (str) the subset
- cls(str) the class

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

```
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

```
ancestor(*Classes)
```

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

# Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

# hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

## Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

 ${\tt resolveword}\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

## Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a <code>TEXTCONTAINER</code>

### xml()

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

# See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

# Return type str

```
___iter___()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.SynsetFeature

```
class pynlpl.formats.folia.SynsetFeature(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.Feature
```

Synset feature, to be used within Sense

# **Method Summary**

init(doc, *args, **kwargs)	Constructor.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

### append(child, *args, **skwargs)  ### context(sizef, placeholder, scope)  ### context(sizef, previousdelimiter, strict,)  ### context(sizef, placeholder, scope)  ### context(sizef, previousdelimiter, strict,)  ### context(sizef, placeholder, scope)  ### context(sizef, previousdelimiter, strict,)  ### context(sizef, previo		ued from previous page
Returns this word in context, [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the left, the current word, and [size] words to the specified in the current word, and [size] words to the infert the content of this class.    Dotain the description associated with the element.	ancestors([Class])	•
context(size[, placeholder, scope])         Returns this word in context. (size] words to the left, the current word, and (size] words to the right words with the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the current word, and (size] words to the right the element.           count(Class], set, recursive, ignore, node])         Like AbstractElement. select (), but instead of returning the elements, it merely counts them.           deepvalidation()         Perform deep validation of this element.           description associated with the element.         Obtain the feature class value of the specific subset.           Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused           finderplaceable sparent[, set])         Get the index at which an element occurs, recursive by default           get_met_adata([key])         Get the index at which an element occurs, recursive by default           get_met_adata([key])         Get the metadata that applies to this element, automatically inherited from parent elements		fectively back-tracing its path to the root element.
the current word, and [size] words to the right  copy(finewdoc, idsuffix))  Make a deep copy of this element and all its children of this element.  Count(Class], set, recursive, ignore, node])  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  deepvalidation()  deepvalidation()  description()  Detain the description associated with the element.  Dotain the description associated with the element.  Dotain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  getmetadata([key])  getmetadata([key])  getmetadata([key])  getmetadata([key])  getmetadata([key])  getmetadata([key])  getmetadata([key])  for the metadata that applies to this element, automatically inherited from parent elements  gettextdelimiter([retaintokenisation])  hasphon([cls, strict, correctionhandling])  hasetax([cls, strict, correctionhandling])  Does this element have phonetic content (of the specified class)  incorrection()  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Petrums the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  priginaltext([cls, correctionhandling])  phoncontent([cls, correctionhandling])  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  Phon([cls, previousdelimiter, strict,])  Get the phonetic content explicitly associated with this element (of the specified class).  Phono([cls, previousdelimiter, strict,])  Get the phonetic content explicitly associated with this element (of the spe		
copy([newdoc, idsuffix])         Make a deep copy of this element and all its children. Generator creating a deep copy of the children of this element.           count(Class[, set, recursive, ignore, node])         Like AbstractElement.select(), but instead of returning the element. Select(), but instead of returning the element, it merely counts them.           deepvalidation()         Perform deep validation of this element.           description()         Obtain the description associated with the element.           findorrectionhandling(cls)         Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused           findreplaceables(parent[, set])         Internal method to find replaceable elements.           getindex(child[, recursive, ignore])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements           gettextdelimiter([retaintokenisation])         Return the text delimiter for this class.           hasphor([cls, strict, correctionhandling])         Does this element have phonetic content (of the specified class)           hastext([cls, strict, correctionhandling])         Does this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json(	<pre>context(size[, placeholder, scope])</pre>	
copychildren([newdoc, idsuffix])         Generator creating a deep copy of the children of this element.           count(Class[, set, recursive, ignore, node])         Like AbstractElement.select(), but instead of returning the elements, it merely counts them.           deepvalidation()         Perform deep validation of this element.           description()         Obtain the description associated with the element.           feat(subset)         Obtain the feature class value of the specific subset.           findcorrectionhandling(cls)         Find the proper correctionhandling given a textelass by looking in the underlying corrections where it is reused           findreplaceables(parent[, set])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements           gettextdelimiter([retaintokenisation])         Return the text delimiter for this class.           Does this element have phonetic content (of the specified class)           hastext([cls, strict, correctionhandling])         Does this element have text (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])		
count(Class[, set, recursive, ignore, node]   Like   AbstractElement. select(), but instead of returning the elements, it merely counts them.		
stead of returning the elements, it merely counts them.  deepvalidation() Perform deep validation of this element.  description() Obtain the description associated with the element.  feat(subset) Obtain the feature class value of the specific subset.  findcorrectionhandling(cls) Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set]) Internal method to find replaceable elements.  getindex(child[, recursive, ignore]) Get the index at which an element occurs, recursive by default!  getmetadata([key]) Get the metadata that applies to this element, automatically inherited from parent elements  gettextdelimiter([retaintokenisation]) Return the text delimiter for this class.  hasphon([cls, strict, correctionhandling]) Does this element have phonetic content (of the specified class)  incorrection() Is this element thave phonetic content (of the specified class)  incorrection() Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems]) Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist]) Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope]) Returns the left context for an element, as a list.  next([Class, scope, reverse]) Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  phon([cls, previouselimiter, strict,]) Get the phonetic content explicitly associated with this element (of the specified class).  Phonocontent([cls, correctionhandling]) This method will be called after an element is added to another and does some checks.  previous([Class, scope]) Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defi	<pre>copychildren([newdoc, idsuffix])</pre>	
them.  description() Perform deep validation of this element.  description() Obtain the description associated with the element.  feat(subset) Obtain the feature class value of the specific subset.  findcorrectionhandling(cls) Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set]) Internal method to find replaceable elements.  getindex(child[, recursive, ignore]) Get the index at which an element occurs, recursive by default!  getmetadata([key]) Get the metadata that applies to this element, automatically inherited from parent elements  gettextdelimiter([retaintokenisation]) Does this element have phonetic content (of the specified class)  hasphon([cls, strict, correctionhandling]) Does this element have phonetic content (of the specified class)  incorrection() Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems]) Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist]) Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  parsexm[(founditems]) Returns the left context for an element, as a list.  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  phon([cls, previousdelimiter, strict,]) Get the phonetic content explicitly associated with this element (of the specified class)  phonocontent([cls, correctionhandling]) Get the phonetic content explicitly associated with this element (of the specified class)  phonocontent([cls, correctionhandling]) Get the phonetic content explicitly associated with this element (of the specified class)  phonocontent([cls, correctionhandling]) Get the phonetic content explicitly associated with this element (of the speci	count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
deepvalidation()         Perform deep validation of this element.           description()         Obtain the description associated with the element.           feat(subset)         Obtain the feature class value of the specific subset.           findcorrectionhandling(cls)         Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused           findreplaceables(parent[, set])         Internal method to find replaceable elements.           getindex(child[, recursive, ignore])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements           getextdelimiter([retaintokenisation])         Does this element have phonetic content (of the specified class)           hastext([cls, strict, correctionhandling])         Does this element have phonetic content (of the specified class)           incorrection()         Is this element parent are text (of the specified class)           insert(index, child, *args, **kwargs)         **semment (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         **Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           jentental parental parental parental parental parental parent		•
description()         Obtain the description associated with the element.           feat(subset)         Obtain the feature class value of the specific subset.           findcorrectionhandling(cls)         Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused           findreplaceables(parent[, set])         Internal method to find replaceable elements.           getindex(child[, recursive, ignore])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements           gettextdelimiter([retaintokenisation])         Return the text delimiter for this class.           basphon([cls, strict, correctionhandling])         Does this element have phonetic content (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           leftcontext(size[, placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it d	deepvalidation()	
Dotain the feature class value of the specific subset.		
Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  getindex(child[, recursive, ignore])  getmetadata([key])  getmetadata([key])  getmetadata([key])  getmetadata([key])  for the metadata that applies to this element, automatically inherited from parent elements  gettextdelimiter([retaintokenisation])  hasphon([cls, strict, correctionhandling])  hasphon([cls, strict, correctionhandling])  ncorrection()  ls this element have phonetic content (of the specified class)  incorrection()  ls this element have text (of the specified class)  incorrection()  ls this element have text (of the specified class)  incorrection()  ls this element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  genument (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  leftcontext(size[, placeholder, scope])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  phon([cls, previousdelimiter, strict,])  parsexmi(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic content explicitly associated with this element (of the specified class)  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		
by looking in the underlying corrections where it is reused		<b>-</b>
reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata[[key])  Get the metadata that applies to this element, automatically inherited from parent elements  gettextdelimiter([retaintokenisation])  hasphon([cls, strict, correctionhandling])  hastext([cls, strict, correctionhandling])  incorrection()  Is this element have phonetic content (of the specified class)  incorrection()  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope])  next([Class, scope, reverse])  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  plane([cls, previousdelimiter, strict,])  phon([cls, previousdelimiter, strict,])  Get the phonetic content explicitly associated with this element (of the specified class).  phonocontent([cls, correctionhandling])  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		
Internal method to find replaceable elements.   getindex(child[, recursive, ignore])		
getindex(child[, recursive, ignore])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements           gettextdelimiter([retaintokenisation])         Return the text delimiter for this class.           hasphon([cls, strict, correctionhandling])         Does this element have phonetic content (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           leftcontext(size[, placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks	findreplaceables(parent[, set])	Internal method to find replaceable elements.
by default!  Get the metadata applies to this element, automatically inherited from parent elements.  Return the text delimiter for this class.  hasphon([cls, strict, correctionhandling])  hastext([cls, strict, correctionhandling])  incorrection()  Is this element have text (of the specified class)  hastext([index, child, *args, **kwargs)  items([founditems])  items([founditems])  pson([attribs, recurse, ignorelist])  next([Class, scope, reverse])  Returns the for context for an element of a correction? If it is, it returns the correction element (not limited to AbstractElement)  serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.  next([Class, scope, reverse])  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  parsexml(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class).  prostappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified class).  Previous([Class, scope])		
matically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Incorrection()  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Insert(index, child, *args, **kwargs)  Items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  Jeon([attribs, recurse, ignorelist])  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Phon([cls, previousdelimiter, strict,])  Postappend()  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Phon(content([cls, correctionhandling]))  Get the phonetic representation associated with this element (of the specified class).  Postappend()  This method will be called after an element is added to another and does some checks.  Previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		by default!
gettextdelimiter([retaintokenisation])         Return the text delimiter for this class.           hasphon([cls, strict, correctionhandling])         Does this element have phonetic content (of the specified class)           hastext([cls, strict, correctionhandling])         Does this element have text (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           leftcontext(size[, placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           originaltext([cls])         Alias for retrieving the original uncorrect text.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class)	getmetadata([key])	Get the metadata that applies to this element, auto-
gettextdelimiter([retaintokenisation])         Return the text delimiter for this class.           hasphon([cls, strict, correctionhandling])         Does this element have phonetic content (of the specified class)           hastext([cls, strict, correctionhandling])         Does this element have text (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           leftcontext(size[, placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           originaltext([cls])         Alias for retrieving the original uncorrect text.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).		
hastext([cls, strict, correctionhandling])       Does this element have text (of the specified class)         incorrection()       Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None         insert(index, child, *args, **kwargs)       Returns a depth-first flat list of all items below this element (not limited to AbstractElement)         json([attribs, recurse, ignorelist])       Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.         leftcontext(size[, placeholder, scope])       Returns the left context for an element, as a list.         next([Class, scope, reverse])       Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.         originaltext([cls])       Alias for retrieving the original uncorrect text.         parsexml(node, doc, **kwargs)       Internal class method used for turning an XML element into an instance of the Class.         phon([cls, previousdelimiter, strict, ])       Get the phonetic representation associated with this element (of the specified class)         phoncontent([cls, correctionhandling])       Get the phonetic content explicitly associated with this element (of the specified class).         postappend()       This method will be called after an element is added to another and does some checks.         previous([Class, scope])       Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	<pre>gettextdelimiter([retaintokenisation])</pre>	
hastext([cls, strict, correctionhandling])         Does this element have text (of the specified class)           incorrection()         Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None           insert(index, child, *args, **kwargs)         Returns a depth-first flat list of all items below this element (not limited to AbstractElement)           json([attribs, recurse, ignorelist])         Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.           leftcontext(size[, placeholder, scope])         Returns the left context for an element, as a list.           next([Class, scope, reverse])         Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.           originaltext([cls])         Alias for retrieving the original uncorrect text.           parsexml(node, doc, **kwargs)         Internal class method used for turning an XML element into an instance of the Class.           phon([cls, previousdelimiter, strict,])         Get the phonetic representation associated with this element (of the specified class)           phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the bou	hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
incorrection()       Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None         insert(index, child, *args, **kwargs)       Eturns ([founditems])         items([founditems])       Returns a depth-first flat list of all items below this element (not limited to AbstractElement)         json([attribs, recurse, ignorelist])       Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.         leftcontext(size[, placeholder, scope])       Returns the left context for an element, as a list.         next([Class, scope, reverse])       Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.         originaltext([cls])       Alias for retrieving the original uncorrect text.         parsexml(node, doc, **kwargs)       Internal class method used for turning an XML element into an instance of the Class.         phon([cls, previousdelimiter, strict, ])       Get the phonetic representation associated with this element (of the specified class).         phoncontent([cls, correctionhandling])       Get the phonetic content explicitly associated with this element (of the specified class).         postappend()       This method will be called after an element is added to another and does some checks.         previous([Class, scope])       Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		ified class)
the Correction element (evaluating to True), otherwise it returns None  insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  next([Class, scope, reverse])  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  Alias for retrieving the original uncorrect text.  parsexml(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	hastext([cls, strict, correctionhandling])	
<pre>insert(index, child, *args, **kwargs)  items([founditems])  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  next([Class, scope, reverse])  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  parsexml(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.</pre>	incorrection()	
insert(index, child, *args, **kwargs)         items([founditems])       Returns a depth-first flat list of all items below this element (not limited to AbstractElement)         json([attribs, recurse, ignorelist])       Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.         leftcontext(size[, placeholder, scope])       Returns the left context for an element, as a list.         next([Class, scope, reverse])       Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.         originaltext([cls])       Alias for retrieving the original uncorrect text.         parsexml(node, doc, **kwargs)       Internal class method used for turning an XML element into an instance of the Class.         phon([cls, previousdelimiter, strict,])       Get the phonetic representation associated with this element (of the specified class)         phoncontent([cls, correctionhandling])       Get the phonetic content explicitly associated with this element (of the specified class).         postappend()       This method will be called after an element is added to another and does some checks.         previous([Class, scope])       Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		the Correction element (evaluating to True), other-
items([founditems])Returns a depth-first flat list of all items below this element (not limited to AbstractElement)json([attribs, recurse, ignorelist])Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.leftcontext(size[, placeholder, scope])Returns the left context for an element, as a list.next([Class, scope, reverse])Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.originaltext([cls])Alias for retrieving the original uncorrect text.parsexml(node, doc, **kwargs)Internal class method used for turning an XML element into an instance of the Class.phon([cls, previousdelimiter, strict,])Get the phonetic representation associated with this element (of the specified class)phoncontent([cls, correctionhandling])Get the phonetic content explicitly associated with this element (of the specified class).postappend()This method will be called after an element is added to another and does some checks.previous([Class, scope])Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		wise it returns None
element (not limited to AbstractElement)  json([attribs, recurse, ignorelist])  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  Alias for retrieving the original uncorrect text.  parsexml(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		
Python dictionary suitable for serialisation to JSON.  leftcontext(size[, placeholder, scope])  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls])  Alias for retrieving the original uncorrect text.  parsexml(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	<pre>items([founditems])</pre>	
leftcontext(size[, placeholder, scope])Returns the left context for an element, as a list.next([Class, scope, reverse])Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.originaltext([cls])Alias for retrieving the original uncorrect text.parsexml(node, doc, **kwargs)Internal class method used for turning an XML element into an instance of the Class.phon([cls, previousdelimiter, strict,])Get the phonetic representation associated with this element (of the specified class)phoncontent([cls, correctionhandling])Get the phonetic content explicitly associated with this element (of the specified class).postappend()This method will be called after an element is added to another and does some checks.previous([Class, scope])Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  originaltext([cls]) Alias for retrieving the original uncorrect text.  parsexml(node, doc, **kwargs) Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,]) Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling]) Get the phonetic content explicitly associated with this element (of the specified class).  postappend() This method will be called after an element is added to another and does some checks.  previous([Class, scope]) Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		Python dictionary suitable for serialisation to JSON.
and if it does not cross the boundary of the defined scope.  originaltext([cls]) Alias for retrieving the original uncorrect text.  parsexml(node, doc, **kwargs) Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,]) Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling]) Get the phonetic content explicitly associated with this element (of the specified class).  postappend() This method will be called after an element is added to another and does some checks.  previous([Class, scope]) Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	<pre>leftcontext(size[, placeholder, scope])</pre>	
scope.  originaltext([cls])  parsexml(node, doc, **kwargs)  Internal class method used for turning an XML element into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	<pre>next([Class, scope, reverse])</pre>	- · · · · · · · · · · · · · · · · · · ·
originaltext([cls])       Alias for retrieving the original uncorrect text.         parsexml(node, doc, **kwargs)       Internal class method used for turning an XML element into an instance of the Class.         phon([cls, previousdelimiter, strict,])       Get the phonetic representation associated with this element (of the specified class)         phoncontent([cls, correctionhandling])       Get the phonetic content explicitly associated with this element (of the specified class).         postappend()       This method will be called after an element is added to another and does some checks.         previous([Class, scope])       Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		and if it does not cross the boundary of the defined
parsexml (node, doc, **kwargs)       Internal class method used for turning an XML element into an instance of the Class.         phon([cls, previousdelimiter, strict,])       Get the phonetic representation associated with this element (of the specified class)         phoncontent([cls, correctionhandling])       Get the phonetic content explicitly associated with this element (of the specified class).         postappend()       This method will be called after an element is added to another and does some checks.         previous([Class, scope])       Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		
ment into an instance of the Class.  phon([cls, previousdelimiter, strict,])  Get the phonetic representation associated with this element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		
element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	parsexml(node, doc, **kwargs)	
element (of the specified class)  phoncontent([cls, correctionhandling])  Get the phonetic content explicitly associated with this element (of the specified class).  postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	phon([cls, previousdelimiter, strict,])	
phoncontent([cls, correctionhandling])         Get the phonetic content explicitly associated with this element (of the specified class).           postappend()         This method will be called after an element is added to another and does some checks.           previous([Class, scope])         Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		
this element (of the specified class).  postappend() This method will be called after an element is added to another and does some checks.  previous([Class, scope]) Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	<pre>phoncontent([cls, correctionhandling])</pre>	
postappend()  This method will be called after an element is added to another and does some checks.  previous([Class, scope])  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.		
to another and does some checks.  **Previous*([Class, scope])*  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.	postappend()	
type and if it does not cross the boundary of the defined scope.		
type and if it does not cross the boundary of the defined scope.	previous([Class, scope])	Returns the previous element, if it is of the specified
fined scope.	-	
Continued on next page		fined scope.
		Continued on next page

Table 87 – continued from previous page

relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
reraxing([includecimaten, extraatinos,])	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace(clind, args, kwargs)	places any existing child element of the same type
	and set.
resolveword(id)	and set.
` /	Determs the right context for an alamost as a list
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1()	Serialises the FoLiA element and all its contents to
	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
2 N.1 7 -1 27	XML.
iter ()	Iterate over all children of this element.
	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Feature'
```

OCCURRENCES\_PER\_SET = 0

OPTIONAL\_ATTRIBS = None

PHONCONTAINER = False

OCCURRENCES = 0

```
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = 'synset'
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = None
Method Details
__init___(doc, *args, **kwargs)
    Constructor.
        Keyword Arguments
            • subset (str) – the subset
            • cls(str) – the class
 __init___(doc, *args, **kwargs)
    Constructor.
        Keyword Arguments
            • subset (str) – the subset
            • cls (str) – the class
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
    This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
    by subclasses for more customised behaviour.
        Parameters
            • parent (AbstractElement) - The element that is being added to
            • set (str or None) - The set
            • raiseexceptions (bool) - Raise an exception if the element can't be added?
```

Returns bool

Raises ValueError

addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

## **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy()* on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### **Returns** str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

#### Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

 $\verb|hastext| (cls='current', strict=True, correction handling=1)|$ 

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.

• correctionhandling – Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

# originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
• node - XML Element (*)-
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

# **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

## resolveword(id)

```
\verb|rightcontext| (size, placeholder=None, scope=None)|
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

## Example:

### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

# speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

## See also:

text() phoncontent() phon()

```
textvalidation(warnonly=None)
```

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

# updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

### xml()

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
iter ()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
    ...
```

### \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.ActorFeature

```
class pynlpl.formats.folia.ActorFeature(doc, *args, **kwargs)
```

Bases: pynlpl.formats.folia.Feature

Actor feature, to be used within Event

# **Method Summary**

init(doc, *args, **kwargs)	Constructor.
<pre>accepts(Class[, raiseexceptions, parentinstance])</pre>	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
(Classif, sed, recarsive, ignore, nead)	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
Timacorrectionnamaring (Cis)	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
geetingen (emidt, recursive, ignore))	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
geemeeaaaea([hejj])	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	
incorrection()	Is this element part of a correction? If it is it returns
	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True) other-
	the Correction element (evaluating to True), other-
insert(index child *aros **kwaros)	
<pre>insert(index, child, *args, **kwargs) items([founditems])</pre>	the Correction element (evaluating to True), otherwise it returns None
<pre>insert(index, child, *args, **kwargs) items([founditems])</pre>	the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
<pre>items([founditems])</pre>	the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
	the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a
<pre>items([founditems])  json([attribs, recurse, ignorelist])</pre>	the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>items([founditems])  json([attribs, recurse, ignorelist])  leftcontext(size[, placeholder, scope])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.
<pre>items([founditems])  json([attribs, recurse, ignorelist])</pre>	the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type
<pre>items([founditems])  json([attribs, recurse, ignorelist])  leftcontext(size[, placeholder, scope])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined
<pre>items([founditems])  json([attribs, recurse, ignorelist])  leftcontext(size[, placeholder, scope])</pre>	the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.  Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type

Table 88 – continued from previous page

	ed from previous page
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
r c c c c c c c c c c c c c c c c c c c	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
previous([Class, scope])	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
reraxing ([includecimaten, extraatulos,])	·
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
<u></u>	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
text([cis, retaintokenisation,])	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
ceaceoncency[cis, confectionnanding])	element (of the specified class).
toutualidation([warmanly])	Run text validation on this element.
textvalidation([warnonly])	
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xml()	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
<u></u>	rent element.
str()	Alias for text ()
<u> </u>	

# **Class Attributes**

ACCEPTED\_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo

```
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Feature'
OCCURRENCES = 0
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = None
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = 'actor'
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = None
```

### **Method Details**

# **Keyword Arguments**

- **subset** (str) the subset
- cls(str) the class

### **Keyword Arguments**

- **subset** (str) the subset
- **cls** (*str*) the class

 $\verb|classmethod| accepts| (\textit{Class}, \textit{raiseexceptions} = \textit{True}, \textit{parentinstance} = \textit{None})$ 

add (child, \*args, \*\*kwargs)

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
\verb|count| (Class, set=None, recursive=True, ignore=True, node=None)|
```

 $Like \ {\tt AbstractElement.select()}, \ but \ instead \ of \ returning \ the \ elements, \ it \ merely \ counts \ them.$ 

### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

# Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

node - XML Element (\*) doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is <code>CorrectionHandling.CURRENT</code>, which will retrieve the corrected/current phonetic content. You can set this to <code>CorrectionHandling.ORIGINAL</code> if you want the phonetic content prior to correction, and <code>CorrectionHandling.EITHER</code> if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

• correctionhandling – Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See  $\verb|AbstractElement.append|$  () for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

oct the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

# **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml**()

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

See also:

AbstractElement.xmlstring() - for direct string output

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
__iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.BegindatetimeFeature

```
class pynlpl.formats.folia.BegindatetimeFeature(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.Feature
```

Begindatetime feature, to be used within Event

# **Method Summary**

init(doc, *args, **kwargs)	Constructor.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
11' 1 CC' ('1 CC [	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
addtoIndex([norecurse])	· · · · · · · · · · · · · · · · · · ·
ancestor(*Classes)	properly added to the index.  Find the most immediate ancestor of the specified
ancestor('Classes)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
ancescors([Class])	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	rectively back tracing its path to the root element.
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
concentioner, practioner, scope])	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
<pre>getmetadata([key])</pre>	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
(5.1	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
(C. 1. 1.11 \( \psi_1 \), \( \psi \psi \)	wise it returns None
insert(index, child, *args, **kwargs)	D. ( 1 4. C. ( 0.4 1'.4 . C. H' 1.1 41'.
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
icon([attribe requires ignoralist])	element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a
<pre>json([attribs, recurse, ignorelist])</pre>	
Loft contout(size[ pleashelder asset))	Python dictionary suitable for serialisation to JSON.
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
	Continued on next page

Table 89 – continued from previous page

	Alice Countries in the series in the series and the series in the series
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
	and set.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
	scop.  Set the text for this element.
settext(text[, cls])	Set the text for this element.
	Set the text for this element.  Retrieves the speaker of the audio or video file asso-
<pre>settext(text[, cls]) speech_speaker()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.
settext(text[, cls])	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file
<pre>settext(text[, cls]) speech_speaker() speech_src()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True
<pre>settext(text[, cls]) speech_speaker() speech_src()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the spec-
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)
<pre>settext(text[, cls]) speech_speaker() speech_src() stricttext([cls])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  Serialises the FoLiA element and all its contents to
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()  xml()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  Serialises the FoLiA element and all its contents to XML.
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,])  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,]))  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()  xml()  xmlstring([pretty_print])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to XML.
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,]))  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()  xml()  xmlstring([pretty_print])iter()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to XML.  Iterate over all children of this element.
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,]))  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()  xml()  xmlstring([pretty_print])</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with strict=True  Recompute textual value based on the text content of the children.  Serialises the FoLiA element and all its contents to XML.  Iterate over all children of this element.  Returns the number of child elements under the cur-
<pre>settext(text[, cls]) speech_speaker()  speech_src()  stricttext([cls]) text([cls, retaintokenisation,]))  textcontent([cls, correctionhandling])  textvalidation([warnonly]) toktext([cls])  updatetext()  xml()  xmlstring([pretty_print])iter()</pre>	Set the text for this element.  Retrieves the speaker of the audio or video file associated with the element.  Retrieves the URL/filename of the audio or video file associated with the element.  Alias for text() with strict=True  Get the text associated with this element (of the specified class)  Get the text content explicitly associated with this element (of the specified class).  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of the children.  Serialises the FoLiA element and all its contents to XML.  Serialises this FoLiA element and all its contents to XML.  Iterate over all children of this element.

# **Class Attributes**

ACCEPTED\_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo

ANNOTATIONTYPE = None

AUTH = True

```
AUTO_GENERATE_ID = False
LABEL = 'Feature'
OCCURRENCES = 0
OCCURRENCES PER SET = 0
OPTIONAL_ATTRIBS = None
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = 'begindatetime'
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = None
Method Details
__init___(doc, *args, **kwargs)
    Constructor.
       Keyword Arguments
           • subset (str) – the subset
           • cls (str) – the class
 __init___(doc, *args, **kwargs)
    Constructor.
       Keyword Arguments
           • subset (str) – the subset
           • cls (str) – the class
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

by subclasses for more customised behaviour.

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

 $Like \ {\tt AbstractElement.select()}, \ but \ instead \ of \ returning \ the \ elements, \ it \ merely \ counts \ them.$ 

#### Returns int

#### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
getindex (child, recursive=True, ignore=True)
```

Get the index at which an element occurs, recursive by default!

### Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

node - XML Element (\*) doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

• cls (str) – The class of the phonetic content to obtain, defaults to current.

• correctionhandling – Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

#### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *text* (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

```
word,text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

# **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

```
See also:
```

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

```
Alias for text () with retaintokenisation=True
```

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xm1**()

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.EnddatetimeFeature

```
class pynlpl.formats.folia.EnddatetimeFeature(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.Feature
```

Enddatetime feature, to be used within Event

# **Method Summary**

init(doc, *args, **kwargs)	Constructor.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
11' 1 CC' ('1 CC [	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
addtoIndex([norecurse])	· · · · · · · · · · · · · · · · · · ·
ancestor(*Classes)	properly added to the index.  Find the most immediate ancestor of the specified
ancestor (Classes)	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
ancescors([Class])	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	rectively back tracing its path to the root element.
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
concentionel, practionel, scope])	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
<pre>getmetadata([key])</pre>	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
(5.1	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
(C. 1. 1.11 \( \psi \) \( \psi \)	wise it returns None
insert(index, child, *args, **kwargs)	D. ( 1 4. C. ( 0.4 1'.4 . C. H' 1.1 41'.
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this
icon([attribe requires ignoralist])	element (not limited to AbstractElement)  Serialises the FoLiA element and all its contents to a
<pre>json([attribs, recurse, ignorelist])</pre>	
loft contout(size[ pleashelder asset))	Python dictionary suitable for serialisation to JSON.
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
	Continued on next page

Table 90 - continued from previous page

	ed from previous page	
originaltext([cls])	Alias for retrieving the original uncorrect text.	
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-	
	ment into an instance of the Class.	
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this	
	element (of the specified class)	
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with	
	this element (of the specified class).	
postappend()	This method will be called after an element is added	
	to another and does some checks.	
previous([Class, scope])	Returns the previous element, if it is of the specified	
	type and if it does not cross the boundary of the de-	
	fined scope.	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
, , ,	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
<u> </u>	places any existing child element of the same type	
	and set.	
resolveword(id)	· · · · · · · · · · · · · · · · · · ·	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
	Correct all parent relations for elements within the	
setparents()	_	
a a to the authority and the state of the st	scop.  Set the text for this element.	
settext(text[, cls])		
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
(5.1.3)	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xml()	Serialises the FoLiA element and all its contents to	
	XML.	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
○ (rī ^—ī .a)	XML.	
iter()	Iterate over all children of this element.	
len ()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text ()	
	11100 101 6626 (/	

# **Class Attributes**

ACCEPTED\_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo

```
ANNOTATIONTYPE = None
AUTH = True
```

AUTO\_GENERATE\_ID = False

LABEL = 'Feature'

OCCURRENCES = 0

OCCURRENCES PER SET = 0

OPTIONAL\_ATTRIBS = None

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = False

REQUIRED\_ATTRIBS = None

REQUIRED\_DATA = None

SETONLY = False

SPEAKABLE = False

SUBSET = 'enddatetime'

TEXTCONTAINER = False

TEXTDELIMITER = None

XLINK = False

XMLTAG = None

### **Method Details**

# **Keyword Arguments**

- **subset** (str) the subset
- cls(str) the class

### **Keyword Arguments**

- **subset** (str) the subset
- cls(str) the class

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

add (child, \*args, \*\*kwargs)

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str or bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

 $Like \ {\tt AbstractElement.select()}, \ but \ instead \ of \ returning \ the \ elements, \ it \ merely \ counts \ them.$ 

### Returns int

#### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

# findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

### Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

node - XML Element (\*) doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

•  ${\tt cls}\,({\it str})$  – The class of the phonetic content to obtain, defaults to current.

• correctionhandling – Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

```
remove (child)
```

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *text* (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

# **Parameters**

- ${\tt cls}\,({\it str})$  The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text() with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

#### **xml**()

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

#### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

### Return type str

```
___iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

# \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# 4.4.3 Alternatives

A key feature of FoLiA is its ability to make explicit alternative annotations, for token annotations, the Alternative (alt) class is used to this end. Alternative annotations are embedded in this structure. This implies the annotation is not authoritative, but is merely an alternative to the actual annotation (if any). Alternatives may typically occur in larger numbers, representing a distribution each with a confidence value (not mandatory). Each alternative is wrapped in its own Alternative element, as multiple elements inside a single alternative are considered dependent and part

of the same alternative. Combining multiple annotation in one alternative makes sense for mixed annotation types, where for instance a pos tag alternative is tied to a particular lemma:

```
alt = word.add(folia.Alternative)
alt.add(folia.PosAnnotation, set='brown-tagset',cls='n',confidence=0.5)
alt = word.add(folia.Alternative) #note that we reassign the variable!
alt.add(folia.PosAnnotation, set='brown-tagset',cls='a',confidence=0.3)
alt = word.add(folia.Alternative)
alt.add(folia.PosAnnotation, set='brown-tagset',cls='v',confidence=0.2)
```

Span annotation elements have a different mechanism for alternatives, for those the entire annotation layer is embedded in a <code>AlternativeLayers</code> element. This element should be repeated for every type, unless the layers it describeds are dependent on it eachother:

```
alt = sentence.add(folia.AlternativeLayers)
layer = alt.add(folia.Entities)
entity = layer.add(folia.Entity, word1,word2,cls="person", confidence=0.3)
```

Because the alternative annotations are **non-authoritative**, normal selection methods such as <code>select()</code> and <code>annotations()</code> will never yield them, unless explicitly told to do so. For this reason, there is an <code>alternatives()</code> method on structure elements, for the first category of alternatives.

In summary, a list of the two relevant classes for alternatives:

Alternative	Element grouping alternative token annotation(s).
AlternativeLayers	Element grouping alternative subtoken annotation(s).

# pynlpl.formats.folia.Alternative

Element grouping alternative token annotation(s).

Multiple alternative elements may occur, each denoting a different alternative. Elements grouped inside an alternative block are considered dependent.

A key feature of FoLiA is its ability to make explicit alternative annotations, for token annotations, this class is used to this end. Alternative annotations are embedded in this structure. This implies the annotation is *not authoritative*, but is merely an alternative to the actual annotation (if any). Alternatives may typically occur in larger numbers, representing a distribution each with a confidence value (not mandatory). Each alternative is wrapped in its an instance of this class, as multiple elements inside a single alternative are considered dependent and part of the same alternative. Combining multiple annotation in one alternative makes sense for mixed annotation types, where for instance a pos tag alternative is tied to a particular lemma.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
	Continued on next page

Table 92 – continued from previous page

	ued from previous page
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
alternatives([Class, set])	Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
annotation(type[, set])	Obtain a single annotation element.
annotations(Class[, set])	Obtain child elements (annotations) of the specified class.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	Apply a correction (TODO: documentation to be written still)
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset) findcorrectionhandling(cls)	Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	-
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasannotation(Class[, set])	Returns an integer indicating whether such as annotation exists, and if so, how many.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
	Continued on next page

T 11 00		•		
Table 92 –	CONTINUED	trom	nrevinis	nage
Table 32	COLITICACA	11 0111	picvious	page

Returns the left context for an element, as a list.  Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element in the context of the Classical Action Classical Acti	
and if it does not cross the boundary of the defined scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML ele-	
scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML ele-	
Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML ele-	
Internal class method used for turning an XML ele-	
<del>_</del>	
ment into an instance of the Class.	
Get the phonetic representation associated with this	
element (of the specified class)	
Get the phonetic content explicitly associated with	
this element (of the specified class).	
This method will be called after an element is added	
to another and does some checks.	
Returns the previous element, if it is of the specified	
type and if it does not cross the boundary of the de-	
fined scope.	
Returns a RelaxNG definition for this element (as an	
XML element (lxml.etree) rather than a string)	
Removes the child element	
Appends a child element like append(), but re-	
places any existing child element of the same type	
and set.	
Returns the right context for an element, as a list.	
Select child elements of the specified class.	
Set a different document.	
Associate a document with this element.	
Correct all parent relations for elements within the	
scop.	
Set the text for this element.	
Retrieves the speaker of the audio or video file asso-	
ciated with the element.  Retrieves the URL/filename of the audio or video file	
associated with the element.	
Alias for text () with strict=True  Get the text associated with this element (of the spec-	
ified class)	
Get the text content explicitly associated with this	
element (of the specified class).	
Run text validation on this element.	
Alias for $text()$ with	
retaintokenisation=True	
Recompute textual value based on the text content of	
the children.	
Serialises the FoLiA element and all its contents to	
XML.	
Serialises this FoLiA element and all its contents to	
XML.	
Iterate over all children of this element.	
Returns the number of child elements under the cur-	
und indicate of china cicinicity under the cut	
rent element.	

# Table 92 – continued from previous page

\_\_str\_\_() Alias for text ()

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractTokenAnnotation'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = False
AUTO_GENERATE_ID = False
LABEL = 'Alternative'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'alt'
Method Details
___init___(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

#### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

# alternatives (Class=None, set=None)

Generator over alternatives, either all or only of a specific annotation type, and possibly restrained also by set.

#### **Parameters**

- **Class** (*class*) The python Class you want to retrieve (e.g. PosAnnotation). Or set to None to select all alternatives regardless of what type they are.
- **set** (str) The set you want to retrieve (defaults to None, which selects irregardless of set)

**Yields** Alternative elements

# ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

# annotation (type, set=None)

Obtain a single annotation element.

A further restriction can be made based on set.

# **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Returns** An element (instance derived from AbstractElement)

### Example:

```
sense = word.annotation(folia.Sense, 'http://some/path/cornetto').cls
```

#### See also:

AllowTokenAnnotation.annotations() AbstractElement.select()

Raises NoSuchAnnotation if no such annotation exists

```
annotations (Class, set=None)
```

Obtain child elements (annotations) of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.

**Yields** Elements (instances derived from AbstractElement)

### Example:

```
for sense in text.annotations(folia.Sense, 'http://some/path/cornetto'):
    ..
```

#### See also:

AbstractElement.select()

# Raises

- AllowTokenAnnotation.annotations()
- NoSuchAnnotation if no such annotation exists

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- **idsuffix** (*str or bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

Apply a correction (TODO: documentation to be written still)

# count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### **Returns** int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate\_id(cls)
```

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### hasannotation (Class, set=None)

Returns an integer indicating whether such as annotation exists, and if so, how many.

See AllowTokenAnnotation.annotations`() for a description of the parameters.

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

items (founditems=[])

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext(cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

**phon** (*cls='current'*, *previousdelimiter=", strict=False*, *correctionhandling=1*)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

# postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

# **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
\verb|select| (Class, set=None, recursive=True, ignore=True, node=None)|
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by COPY ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text (str) The text
- cls (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext(cls='current')
    Alias for text () with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhandling=1, normalize spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- previousdelimiter (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- strict (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't
- normalize spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

# See also:

```
AbstractElement.xmlstring() - for direct string output
```

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:

__len__()
Returns the number of child elements under the current element.

__str__()
Alias for text()
```

# pynlpl.formats.folia.AlternativeLayers

```
class pynlpl.formats.folia.AlternativeLayers (doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Element grouping alternative subtoken annotation(s). Multiple altlayers elements may occur, each denoting a different alternative. Elements grouped inside an alternative block are considered dependent.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
1	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
	Continued on next page

Table 93 –	continued	from	previous	page

Table 93 – continued from previous page		
getmetadata([key])	Get the metadata that applies to this element, auto-	
	matically inherited from parent elements	
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.	
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-	
	ified class)	
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)	
incorrection()	Is this element part of a correction? If it is, it returns	
	the Correction element (evaluating to True), other-	
	wise it returns None	
<pre>insert(index, child, *args, **kwargs)</pre>		
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this	
	element (not limited to AbstractElement)	
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a	
	Python dictionary suitable for serialisation to JSON.	
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.	
next([Class, scope, reverse])	Returns the next element, if it is of the specified type	
<del>-</del>	and if it does not cross the boundary of the defined	
	scope.	
originaltext([cls])	Alias for retrieving the original uncorrect text.	
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-	
	ment into an instance of the Class.	
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this	
	element (of the specified class)	
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with	
	this element (of the specified class).	
postappend()	This method will be called after an element is added	
	to another and does some checks.	
previous([Class, scope])	Returns the previous element, if it is of the specified	
1 3/	type and if it does not cross the boundary of the de-	
	fined scope.	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
-	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
(C) / (C) -	ified class)	
	Continued on next page	
	2 5 dad on nom pago	

Table 93 – continued from previous page

textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text ()	

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractAnnotationLayer'>, <class 'pynlp
ANNOTATIONTYPE = None
AUTH = False
AUTO_GENERATE_ID = False
LABEL = 'Alternative Layers'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'altlayers'
```

### **Method Details**

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

# addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- **idsuffix** (*str or bool*) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

### count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

Returns int

# deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# $\verb"getindex" (child, recursive=True, ignore=True)"$

Get the index at which an element occurs, recursive by default!

Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

# incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

## leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### **Parameters**

```
    node - XML Element (*) -
    doc - Document (*) -
```

Returns An instance of the current Class.

phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text() textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

# See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

## **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

 $\begin{tabular}{ll} \textbf{classmethod relaxng} (include children = True, & extraattribs = None, & extraelements = None, & original class = None) & extraelements = None, & original class = None, & or$ 

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

#### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

# setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

**text** (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

# textvalidation (warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

# Returns bool

# toktext (cls='current')

Alias for text() with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

# 4.4.4 Corrections

Corrections are one of the most complex annotation types in FoLiA. Corrections can be applied not just over text, but over any type of structure annotation, token annotation or span annotation. Corrections explicitly preserve the original, and recursively so if corrections are done over other corrections.

Despite their complexity, the library treats correction transparently. Whenever you query for a particular element, and it is part of a correction, you get the corrected version rather than the original. The original is always *non-authoritative* and normal selection methods will ignore it.

If you want to deal with correction, you have to explicitly handle the *Correction* element. If an element is part of a correction, its *AbstractElement.incorrection()* method will give the correction element, if not, it will return None:

Corrections themselves carry a class too, indicating the type of correction (defined by the set used and not by FoLiA).

Besides Correction.original(), corrections distinguish three other types, Correction.new() (the corrected version), Correction.current() (the current uncorrected version) and Correction.suggestions() (a suggestion for correction), the former two and latter two usually form pairs, current() and new() can never be used together. Of suggestions (index) there may be multiple, hence the index argument. These return, respectively, instances of Original, folia.New, folia.Current and folia.Suggestion.

Adding a correction can be done explicitly:

```
wrongpos = word.annotation(folia.PosAnnotation)
word.add(folia.Correction, folia.New(doc, folia.PosAnnotation(doc, cls="n")) , folia.
→Original(doc, wrongpos), cls="misclassified")
```

Let's settle for a suggestion rather than an actual correction:

In some instances, when correcting text or structural elements, *New* may be empty, which would correspond to an *deletion*. Similarly, *Original* may be empty, corresponding to an *insertion*.

The use of *Current* is reserved for use with structure elements, such as words, in combination with suggestions. The structure elements then have to be embedded in *Current*. This situation arises for instance when making suggestions for a merge or split.

Here is a list of all relevant classes for corrections:

Correction	Corrections are one of the most complex annotation
Collection	Corrections are one of the most complex annotation
	types in FoLiA.
Current	Used in the context of Correction to encapsulate the
	currently authoritative annotations.
ErrorDetection	The ErrorDetection element is used to signal the pres-
	ence of errors in a structural element.
New	
Original	Used in the context of Correction to encapsulate the
	original annotations <i>prior</i> to correction.
Suggestion	Suggestions are used in the context of Correction,
	but rather than provide an authoritative correction, it in-
	stead offers a suggestion for correction.

# pynlpl.formats.folia.Correction

Corrections are one of the most complex annotation types in FoLiA. Corrections can be applied not just over text, but over any type of structure annotation, token annotation or span annotation. Corrections explicitly preserve the original, and recursively so if corrections are done over other corrections.

Despite their complexity, the library treats correction transparently. Whenever you query for a particular element, and it is part of a correction, you get the corrected version rather than the original. The original is always *non-authoritative* and normal selection methods will ignore it.

### This class takes four classes as children, that in turn encapsulate the actual annotations:

- New Encapsulates the newly corrected annotation(s)
- Original Encapsulated the old original annotation(s)
- Current Encapsulates the current authoritative annotation(s)
- Suggestions Encapsulates the annotation(s) that are a non-authoritative suggestion for correction

# **Method Summary**

Tests whether a new element of this class can be
added to the parent.

Continued on next page

Table 95 -	continued	from	previous	page
14510 00	00111111000		piotioac	Page

	ued from previous page
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
<pre>append(child, *args, **kwargs) context(size[, placeholder, scope])</pre>	See AbstractElement.append()  Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this element.
correct(**kwargs)	
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
<pre>current([index])</pre>	Get the current authoritative annotation (used with suggestions in a structural context)
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	•
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	See AbstractElement. gettextdelimiter()
hascurrent([allowempty])	Does the correction record the current authoritative annotation (needed only in a structural context when suggestions are proposed)
hasnew([allowempty])	Does the correction define new corrected annotations?
hasoriginal([allowempty])	Does the correction record the old annotations prior to correction?
hasphon([cls, strict, correctionhandling])	See AbstractElement.hasphon()
hassuggestions([allowempty])	Does the correction propose suggestions for correction?
hastext([cls, strict, correctionhandling])	See AbstractElement.hastext()
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	Who it fetaling facility
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
	Continued on next page

json([attribs, recurse, ignorelist])	serialises the FoLiA element and all its contents to a	
([uttros, recurse, ignorense])	Python dictionary suitable for serialisation to JSON.	
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.	
new([index])	Get the new corrected annotation.	
next([Class, scope, reverse])	Returns the next element, if it is of the specified typ	
	and if it does not cross the boundary of the defined	
	scope.	
original([index])	Get the old annotation prior to correction.	
originaltext([cls])	Alias for retrieving the original uncorrect text.	
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-	
	ment into an instance of the Class.	
phon([cls, previousdelimiter, strict,])	See AbstractElement.phon()	
<pre>phoncontent([cls, correctionhandling])</pre>	See AbstractElement.phoncontent()	
postappend()	This method will be called after an element is added	
	to another and does some checks.	
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified	
	type and if it does not cross the boundary of the de-	
	fined scope.	
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
7/21)	and set.	
resolveword(id)	Determent the wight content for an element of a list	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
setdoc(newdoc) setdocument(doc)	Set a different document.  Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
setparents()	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
Specen_Speaker()	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
suggestions([index])	Get suggestions for correction.	
text([cls, retaintokenisation,])	See AbstractElement.text()	
LEAL(ICIS, ICIAIIIIUNCIIISAIIUII,   )		
	See AbstractElement.textcontent()	
textcontent([cls, correctionhandling]) textvalidation([warnonly])		
textcontent([cls, correctionhandling])	See AbstractElement.textcontent()	
<pre>textcontent([cls, correctionhandling]) textvalidation([warnonly])</pre>	See AbstractElement.textcontent() Run text validation on this element.	
<pre>textcontent([cls, correctionhandling]) textvalidation([warnonly])</pre>	See AbstractElement.textcontent() Run text validation on this element. Alias for text() with	
<pre>textcontent([cls, correctionhandling]) textvalidation([warnonly]) toktext([cls])</pre>	See AbstractElement.textcontent() Run text validation on this element. Alias for text() with retaintokenisation=True	
<pre>textcontent([cls, correctionhandling]) textvalidation([warnonly]) toktext([cls])</pre>	See AbstractElement.textcontent()  Run text validation on this element.  Alias for text() with retaintokenisation=True  Recompute textual value based on the text content of	
textcontent([cls, correctionhandling]) textvalidation([warnonly]) toktext([cls])  updatetext()  xml([attribs, elements, skipchildren])	See AbstractElement.textcontent() Run text validation on this element. Alias for text() with retaintokenisation=True Recompute textual value based on the text content of the children. Serialises the FoLiA element and all its contents to XML.	
textcontent([cls, correctionhandling]) textvalidation([warnonly]) toktext([cls]) updatetext()	See AbstractElement.textcontent() Run text validation on this element. Alias for text() with retaintokenisation=True Recompute textual value based on the text content of the children. Serialises the FoLiA element and all its contents to	

XML.

Iterate over all children of this element.

4.4. Higher-Order Annotations

\_iter\_\_()

Continued on next page

Table 95 – continued from previous page

len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.
ANNOTATIONTYPE = 16
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Correction'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = True
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'correction'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
    Tests whether a new element of this class can be added to the parent.
```

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

## ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors (Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

# **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

# copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

### current (index=None)

Get the current authoritative annotation (used with suggestions in a structural context)

This returns only one annotation if multiple exist, use *index* to select another in the sequence.

**Returns** an annotation element (AbstractElement)

Raises NoSuchAnnotation

# deepvalidation()

Perform deep validation of this element.

Raises DeepValidationError

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### **Returns** str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

# getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

See AbstractElement.gettextdelimiter()

# hascurrent (allowempty=False)

Does the correction record the current authoritative annotation (needed only in a structural context when suggestions are proposed)

#### hasnew (allowempty=False)

Does the correction define new corrected annotations?

### hasoriginal (allowempty=False)

Does the correction record the old annotations prior to correction?

```
\verb|hasphon|| (cls='current', strict=True, correction handling=1)|
```

```
See AbstractElement.hasphon()
```

#### hassuggestions (allowempty=False)

Does the correction propose suggestions for correction?

```
hastext (cls='current', strict=True, correctionhandling=1)
```

```
See AbstractElement.hastext()
```

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

## Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

#### **new** (*index=None*)

Get the new corrected annotation.

This returns only one annotation if multiple exist, use *index* to select another in the sequence.

**Returns** an annotation element (AbstractElement)

```
Raises NoSuchAnnotation
```

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

### original (index=None)

Get the old annotation prior to correction.

This returns only one annotation if multiple exist, use *index* to select another in the sequence.

**Returns** an annotation element (AbstractElement)

Raises NoSuchAnnotation

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

```
\begin{tabular}{ll} \bf phon~(cls='current',previous delimiter=",strict=False,correction handling=1)\\ \bf See~AbstractElement.phon~() \end{tabular}
```

```
phoncontent (cls='current', correctionhandling=1)
```

See AbstractElement.phoncontent()

## postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set
  to True to constrain to the same class as that of the current instance, set to None to not
  constrain at all
- scope (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

# remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

#### resolveword(id)

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

# Example:

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument(doc)

Associate a document with this element.

```
\textbf{Parameters doc} \ (\textit{Document}) - A \ document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

• text (str) - The text

• **cls** (*str*) – The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

### suggestions (index=None)

Get suggestions for correction.

**Yields** Suggestion element that encapsulate the suggested annotations (if index is None, default)

**Returns** a Suggestion element that encapsulate the suggested annotations (if index is set)

Raises IndexError

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhandling=1, normalize\_spaces=False)
See AbstractElement.text()

```
textcontent (cls='current', correctionhandling=1)
```

See AbstractElement.textcontent()

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

# See also:

AbstractElement.xmlstring() - for direct string output

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

# Return type str

```
__iter__()
```

Iterate over all children of this element.

# Example:

```
for annotation in word:
    ...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.Current

```
class pynlpl.formats.folia.Current(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractCorrectionChild
```

Used in the context of *Correction* to encapsulate the currently authoritative annotations.

Needed only when suggestions for correction are proposed (Suggestion) for structural elements.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
	Continued on next page

Table 96 – continued from previous page

	led from previous page
correct(**kwargs)	
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is reused
<pre>findreplaceables(parent[, set])</pre>	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
-	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
1 (1 , ,	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
items([founditems])	Returns a depth-first flat list of <i>all</i> items below this
z come ([roundrems])	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
Joon ([action, rection, ignoremotif)	Python dictionary suitable for serialisation to JSON.
leftcontext(size[, placeholder, scope])	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
mexe([etass, scope, reverse])	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
parseami (node, doc, kwaigs)	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	
	Get the phonetic representation associated with this
priori([eis, previousaerimiter, strict,])	Get the phonetic representation associated with this
-	element (of the specified class)
-	element (of the specified class)  Get the phonetic content explicitly associated with
phoncontent([cls, correctionhandling])	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).
phoncontent([cls, correctionhandling])	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added
phoncontent([cls, correctionhandling])  postappend()	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.
phoncontent([cls, correctionhandling])  postappend()	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified
phoncontent([cls, correctionhandling])  postappend()	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the de-
phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but re-
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but replaces any existing child element of the same type
<pre>phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])  relaxng([includechildren, extraattribs,])  remove(child)  replace(child, *args, **kwargs)  resolveword(id)</pre>	element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)  Removes the child element  Appends a child element like append(), but re-

Table 96 – continued from previous page

	1 1 5		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.		
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.		
setdoc(newdoc)	Set a different document.		
setdocument(doc)	Associate a document with this element.		
setparents()	Correct all parent relations for elements within the		
	scop.		
settext(text[, cls])	Set the text for this element.		
speech_speaker()	Retrieves the speaker of the audio or video file asso-		
	ciated with the element.		
speech_src()	Retrieves the URL/filename of the audio or video file		
	associated with the element.		
stricttext([cls])	Alias for text() with strict=True		
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-		
	ified class)		
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this		
	element (of the specified class).		
textvalidation([warnonly])	Run text validation on this element.		
toktext([cls])	Alias for text() with		
	retaintokenisation=True		
updatetext()	Recompute textual value based on the text content of		
	the children.		
xml([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to		
	XML.		
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to		
	XML.		
iter()	Iterate over all children of this element.		
len()	Returns the number of child elements under the cur-		
	rent element.		
str()	Alias for text ()		

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractSpanAnnotation'>, <class 'pynlpl
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
```

OCCURRENCES = 1

OCCURRENCES\_PER\_SET = 0

OPTIONAL\_ATTRIBS = None

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = True

REQUIRED\_ATTRIBS = None

REQUIRED\_DATA = None

SETONLY = False

 $\verb|classmethod| addable| (parent, set=None, raise exceptions=True)|$ 

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

#### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

```
addtoindex (norecurse=[])
```

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

```
ancestor (*Classes)
```

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

#### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

# ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
correct (**kwargs)
```

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

```
generate_id(cls)
```

getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

```
getmetadata (key=None)
```

Get the metadata that applies to this element, automatically inherited from parent elements

```
gettextdelimiter (retaintokenisation=False)
```

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

```
hasphon (cls='current', strict=True, correctionhandling=1)
```

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Returns bool

hastext(cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

# **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

#### json (attribs=None, recurse=True, ignorelist=False)

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

# **Parameters**

```
    node - XML Element (*) -
    doc - Document (*) -
```

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.

- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike phon (), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy ()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

# speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.

- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

# Returns bool

# toktext (cls='current')

Alias for text () with retaintokenisation=True

## updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

AbstractElement.xmlstring() - for direct string output

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

```
__len__()
```

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.ErrorDetection

```
class pynlpl.formats.folia.ErrorDetection(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractExtendedTokenAnnotation
```

The ErrorDetection element is used to signal the presence of errors in a structural element.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	See AbstractElement.append()
-	Continued on payt page

Continued on next page

Table 97 – continued from previous page

	ed from previous page
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
iinacorrectionnanaring(cis)	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
v	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
([	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
good ([attaios, recurse, ignoremot])	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
next([Class, scope, reverse])	and if it does not cross the boundary of the defined
	•
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
· · · · · · · · · · · · · · · · · · ·	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
<del>-</del>	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
J.(L. 1977)	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
	Continued on next page
	Sommuca on next page

Table 97 – continued from previous page

	ued from previous page	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
<pre>rightcontext(size[, placeholder, scope])</pre>	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
•	XML.	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text ()	

# **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 17

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Error Detection'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 10, 11)

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = False
```

```
REQUIRED\_ATTRIBS = (1,)
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'errordetection'
Method Details
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
             from. Not instances!
```

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

See AbstractElement.append()

# context (size, placeholder=None, scope=None)

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

# description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

# feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

# classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

# getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

# hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

### Return type str

```
___iter___()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
```

### \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

# pynlpl.formats.folia.New

```
class pynlpl.formats.folia.New(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractCorrectionChild
```

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
	Continued on next page

Continued on next page

Table 98 – continued from previous page

	ued from previous page
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
o o <sub>1</sub> - 1 o o o o o o (t-o o o o o o o o o o o o o o o o o o o	element.
correct(**kwargs)	V. C.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
Count (Class[, set, recursive, ignore, node])	
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	1
getindex(child[, recursive, ignore])	Get the index at which an element occurs, recursive
gooding, recursive, ignorej)	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
getmetadata([Key])	
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
insert(index, child, *args, **kwargs)	
items([founditems])	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
jeon([attitos, recurse, ignorenst])	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
	Returns the next element, if it is of the specified type
next([Class, scope, reverse])	- · · · · · · · · · · · · · · · · · · ·
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this
	element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with
1	this element (of the specified class).
postappend()	This method will be called after an element is added
postappenu()	to another and does some checks.
n novi ou o([Class sages])	
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
	Continued on next page

Table 98 – continued from previous page

relaxng([includechildren, extraattribs,])       Returns a RelaxNG definition for this element (XML element (lxml.etree) rather than a string)         remove(child)       Removes the child element         replace(child, *args, **kwargs)       Appends a child element like append(), but places any existing child element of the same and set.         resolveword(id)       Returns the right context for an element, as a list select(Class[, set, recursive, ignore, node])       Returns the right context for an element, as a list select child elements of the specified class.         setdoc(newdoc)       Set a different document.	ıt re- type		
remove(child)       Removes the child element         replace(child, *args, **kwargs)       Appends a child element like append(), but places any existing child element of the same and set.         resolveword(id)       Returns the right context for an element, as a list select(Class[, set, recursive, ignore, node])         select child elements of the specified class.         setdoc(newdoc)       Set a different document.	type		
replace(child, *args, **kwargs)       Appends a child element like append(), but places any existing child element of the same and set.         resolveword(id)       Returns the right context for an element, as a list select(Class[, set, recursive, ignore, node])       Select child elements of the specified class.         setdoc(newdoc)       Set a different document.	type		
places any existing child element of the same and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  setdoc(newdoc)  Returns the right context for an element, as a list select child elements of the specified class.  Set a different document.	type		
and set.  resolveword(id)  rightcontext(size[, placeholder, scope])  select(Class[, set, recursive, ignore, node])  setdoc(newdoc)  Returns the right context for an element, as a list select child elements of the specified class.  Set a different document.			
resolveword(id)rightcontext(size[, placeholder, scope])Returns the right context for an element, as a listselect(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.			
rightcontext(size[, placeholder, scope])Returns the right context for an element, as a listselect(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.			
select(Class[, set, recursive, ignore, node])Select child elements of the specified class.setdoc(newdoc)Set a different document.	SE.		
setdoc(newdoc) Set a different document.			
setdocument (doc) Associate a document with this element.			
setparents() Correct all parent relations for elements within	n the		
scop.			
settext(text[, cls]) Set the text for this element.			
speech_speaker() Retrieves the speaker of the audio or video file	asso-		
ciated with the element.			
speech_src() Retrieves the URL/filename of the audio or vide	Retrieves the URL/filename of the audio or video file		
associated with the element.			
stricttext([cls]) Alias for text() with strict=True			
text([cls, retain to kenisation,]) Get the text associated with this element (of the	spec-		
ified class)			
textcontent([cls, correctionhandling]) Get the text content explicitly associated with	this		
element (of the specified class).			
textvalidation([warnonly]) Run text validation on this element.			
toktext([cls]) Alias for text()	with		
retaintokenisation=True			
updatetext() Recompute textual value based on the text conte	ent of		
the children.			
xm1([attribs, elements, skipchildren]) Serialises the FoLiA element and all its content	its to		
XML.			
xmlstring([pretty_print]) Serialises this FoLiA element and all its content	its to		
XML.			
iter() Iterate over all children of this element.			
len() Returns the number of child elements under the	cur-		
	rent element.  Alias for text ()		
str () Alias for text ()			

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractSpanAnnotation'>, <class 'pynlpl
ANNOTATIONTYPE = None
AUTH = True</pre>
```

AUTO\_GENERATE\_ID = False

OCCURRENCES = 1

OCCURRENCES\_PER\_SET = 0

OPTIONAL\_ATTRIBS = None

PHONCONTAINER = False

PRIMARYELEMENT = True

```
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'new'
Method Details
__init__ (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
___init___(doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
             from. Not instances!
```

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
correct (**kwargs)
```

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

## Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

#### generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike *phon()*, this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

 $\verb|hastext| (cls='current', strict=True, correction handling=1)$ 

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

#### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

### Return type str

```
__iter__()
```

Iterate over all children of this element.

## Example:

```
for annotation in word:
```

## \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.Original

```
class pynlpl.formats.folia.Original(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractCorrectionChild
```

Used in the context of Correction to encapsulate the original annotations prior to correction.

## **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

Continued on next page

Table 99 – continued from previous pag	Jе
----------------------------------------	----

	ued from previous page
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
Timacorrectionnanaring(cis)	by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
<pre>incorrection()</pre>	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
<pre>phoncontent([cls, correctionhandling])</pre>	Get the phonetic content explicitly associated with this element (of the specified class).
postappend()	This method will be called after an element is added to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.
	Continued on next page

Table 99 – continued from previous page

relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an		
reraxing ([includecimaten, extraatinos,])	XML element (lxml.etree) rather than a string)		
remove(child)	Removes the child element		
replace(child, *args, **kwargs)	Appends a child element like append(), but re-		
repraed (mid, digo, kwaigo)	places any existing child element of the same type		
	and set.		
resolveword(id)			
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.		
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.		
setdoc(newdoc)	Set a different document.		
setdocument(doc)	Associate a document with this element.		
setparents()	Correct all parent relations for elements within the		
	scop.		
settext(text[, cls])	Set the text for this element.		
speech_speaker()	Retrieves the speaker of the audio or video file asso-		
	ciated with the element.		
speech_src()	Retrieves the URL/filename of the audio or video file		
	associated with the element.		
stricttext([cls])	Alias for text () with strict=True		
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-		
	ified class)		
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this		
	element (of the specified class).		
textvalidation([warnonly])	Run text validation on this element.		
toktext([cls])	Alias for text() with		
	retaintokenisation=True		
updatetext()	Recompute textual value based on the text content of		
	the children.		
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to		
	XML.		
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to		
	XML.		
iter()	Iterate over all children of this element.		
len()	Returns the number of child elements under the cur-		
	rent element.		
str()	Alias for text ()		

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractSpanAnnotation'>, <class 'pynlpl
ANNOTATIONTYPE = None
AUTH = False
AUTO_GENERATE_ID = False</pre>
```

OCCURRENCES\_PER\_SET = 0

OCCURRENCES = 1

OPTIONAL\_ATTRIBS = None

PHONCONTAINER = False

PRIMARYELEMENT = True

```
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'original'
Method Details
__init__ (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
___init___(doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
             from. Not instances!
```

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

## hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

## Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

## textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

## See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

Returns a string with XML representation for this element and all its children

### Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

```
for annotation in word:
```

### \_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
Alias for text()
```

## pynlpl.formats.folia.Suggestion

```
class pynlpl.formats.folia.Suggestion(doc, *args, **kwargs)
     Bases: pynlpl.formats.folia.AbstractCorrectionChild
```

Suggestions are used in the context of *Correction*, but rather than provide an authoritative correction, it instead offers a suggestion for correction.

### **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
	Continued on next page

Table 100 – continued	from	prev	ous pa	ge
	7		' 1 1'	11

	ued from previous page
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
	element.
count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
Tinacorrectionnanaring(cis)	
	by looking in the underlying corrections where it is
	reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
generate_id(cls)	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
V	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
([])	element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
JSOM [attribs, recurse, ignorenst])	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
onicinal+out([ala])	Scope.  A lies for retrieving the original uncorrect tout
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
	Continued on next page
	Continued on next page

Table 100 – continued from previous page

relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an	
reraxing([includecimaten, extraatinos,])	XML element (lxml.etree) rather than a string)	
remove(child)	Removes the child element	
replace(child, *args, **kwargs)	Appends a child element like append(), but re-	
	places any existing child element of the same type	
	and set.	
resolveword(id)		
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text () with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
	XML.	
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to	
	XML.	
iter()	Iterate over all children of this element.	
len()	Returns the number of child elements under the cur-	
	rent element.	
str()	Alias for text()	

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AbstractSpanAnnotation'>, <class 'pynlpl
ANNOTATIONTYPE = None
AUTH = False
AUTO_GENERATE_ID = False
OCCURRENCES = 0
```

OPTIONAL\_ATTRIBS = (0, 2, 3, 5, 4)
PHONCONTAINER = False

PRIMARYELEMENT = True

OCCURRENCES\_PER\_SET = 0

```
PRINTABLE = True
REQUIRED ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = True
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'suggestion'
Method Details
__init__ (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
___init___(doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
             from. Not instances!
```

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

## description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

## Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

Returns str or list

### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

### generate\_id(cls)

### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

## hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

```
A call to text () with correctionhandling=CorrectionHandling.ORIGINAL
```

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

## **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use <code>AbstractElement.append()</code> if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

 ${\tt resolveword}\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

#### Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

### setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
\begin{tabular}{ll} \textbf{text} (cls='current', retaintokenisation=False, previous delimiter='', strict=False, correction handling=1, normalize\_spaces=False) \\ \end{tabular}
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

# textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

# See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain to kenisation=True

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

**Returns** an lxml.etree.Element

### See also:

AbstractElement.xmlstring() - for direct string output

## xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

### Example:

Alias for text()

```
for annotation in word:

...

__len__()

Returns the number of child elements under the current element.

__str__()
```

# 4.4.5 Alignments

Alignments are used to make reference to external documents. It concerns references as annotation rather than references which are explicitly part of the text, such as hyperlinks and Reference.

The following elements are relevant for alignments:

Alignment	The Alignment element is a form of higher-order anno-
	tation taht is used to point to an external resource.
AlignReference	The AlignReference element is used to point to specific
	elements inside the aligned source.

# pynlpl.formats.folia.Alignment

```
class pynlpl.formats.folia.Alignment(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

The Alignment element is a form of higher-order annotation taht is used to point to an external resource.

It concerns references as annotation rather than references which are explicitly part of the text, such as hyperlinks and Reference.

Inside the Alignment element, the AlignReference element may be used to point to specific elements

(multiple denotes a span).

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be added to the parent.
<pre>addidsuffix(idsuffix[, recursive])</pre>	Appends a suffix to this element's ID, and optionally to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
context(size[, placeholder, scope])	Returns this word in context, {size} words to the left, the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but instead of returning the elements, it merely counts them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused
findreplaceables(parent[, set])	Internal method to find replaceable elements.
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive by default!
getmetadata([key])	Get the metadata that applies to this element, automatically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the specified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), other- wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
	Continued on next page

 $t \in xt([cls, retaintokenisation, ...])$ 

textvalidation([warnonly])

xml([attribs, elements, skipchildren])

xmlstring([pretty\_print])

toktext([cls])

updatetext()

\_iter\_\_()

len\_\_()

textcontent([cls, correctionhandling])

next([Class, scope, reverse])	Returns the next element, if it is of the specified type
	and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
<pre>parsexml(node, doc, **kwargs)</pre>	Internal class method used for turning an XML ele-
	ment into an instance of the Class.
<pre>phon([cls, previousdelimiter, strict,])</pre>	Get the phonetic representation associated with this
	element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
<pre>previous([Class, scope])</pre>	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the de-
	fined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an
	XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
	places any existing child element of the same type
7 ([1	and set.
resolve([documents])	
resolveword(id)	D 4 4 2 14 4 6 1 4 12 4
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
	scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
speeci_sic()	
	associated with the element.
stricttext([cls])	associated with the element.  Alias for text() with strict=True

ified class)

the children.

rent element.

Alias

Continued on next page

Get the text associated with this element (of the spec-

Get the text content explicitly associated with this

Recompute textual value based on the text content of

Serialises the FoLiA element and all its contents to

Serialises this FoLiA element and all its contents to

Returns the number of child elements under the cur-

text()

element (of the specified class).
Run text validation on this element.

for

retaintokenisation=True

Iterate over all children of this element.

with

# Table 102 – continued from previous page

 $\_str\_()$  Alias for text ()

### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.AlignReference'>, <class 'pynlpl.formats
ANNOTATIONTYPE = 26
AUTH = True
AUTO_GENERATE_ID = False
LABEL = 'Alignment'
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = True
XMLTAG = 'alignment'
Method Details
___init___(doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

### addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor (\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

## Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

## ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- **idsuffix** (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

#### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

# findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

## getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is <code>CorrectionHandling.CURRENT</code>, which will retrieve the corrected/current phonetic content. You can set this to <code>CorrectionHandling.ORIGINAL</code> if you want the phonetic content prior to correction, and <code>CorrectionHandling.EITHER</code> if you don't care.

Returns bool

```
hastext (cls='current', strict=True, correctionhandling=1)
```

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

## Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

# Example:

```
import json
json.dumps(word.json())
```

### Returns dict

### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

#### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

node - XML Element (\*) doc - Document (\*) -

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)
```

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to phon(). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is <code>CorrectionHandling.CURRENT</code>, which will retrieve the corrected/current phonetic content. You can set this to <code>CorrectionHandling.ORIGINAL</code> if you want the phonetic content prior to correction, and <code>CorrectionHandling.EITHER</code> if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

# See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

```
phoncontent (cls='current', correctionhandling=1)
```

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

• correctionhandling – Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

## See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng(includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

```
remove (child)
```

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- **alternative** (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See  $\verb|AbstractElement.append|$  () for more information and all parameters.

```
resolve (documents=None)
```

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*) Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off
   AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

Yields Elements (instances derived from AbstractElement)

### Example:

### setdoc (newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# $\verb"setdocument" (doc)$

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

- text(str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

 $\begin{tabular}{ll} \textbf{text} (cls='current', retaintokenisation=False, previous delimiter=", strict=False, correction handling=1, normalize\_spaces=False) \end{tabular}$ 

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

```
word.text()
```

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchText – if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

# Parameters

- cls (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the

corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

```
Alias for text () with retaintokenisation=True
```

#### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a  ${\tt TEXTCONTAINER}$ 

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
__str__()
```

Alias for text ()

# pynlpl.formats.folia.AlignReference

class pynlpl.formats.folia.AlignReference(doc, \*args, \*\*kwargs)
 Bases: pynlpl.formats.folia.AbstractElement

The AlignReference element is used to point to specific elements inside the aligned source.

It is used with Alignment which is responsible for pointing to the external resource.

# **Method Summary**

added to the parent.  addidsuffix(idsuffix[, recursive])  Appends a suffix to this element's ID, and optionally to all child IDs as well.  Addtoindex([norecurse])  Makes sure this element (and all subelements), are properly added to the index.  ancestor(*Classes)  Find the most immediate ancestor of the specified type, multiple classes may be specified.  ancestors([Class])  Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.  append(child, *args, **kwargs)  context(size[, placeholder, scope])  Returns this word in context, {size} words to the left the current word, and {size} words to the right copy([newdoc, idsuffix])  Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])  Generator creating a deep copy of the children of this element.  count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  deepvalidation()  Perform deep validation of this element.  description()  Obtain the description associated with the element.  feat(subset)  findcorrectionhandling(cls)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!	init(doc, *args, **kwargs)	Initialize self.
add(child, *args, **kwargs)         addable(parent[, set, raiseexceptions])       Tests whether a new element of this class can be added to the parent.         addidsuffix(idsuffix[, recursive])       Appends a suffix to this element's ID, and optionally to all child IDs as well.         addtoindex([norecurse])       Makes sure this element (and all subelements), are properly added to the index.         ancestor(*Classes)       Find the most immediate ancestor of the specified type, multiple classes may be specified.         ancestors([Class])       Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.         append(child, *args, ***kwargs)       Returns this word in context, {size} words to the left the current word, and {size} words to the right         copy([newdoc, idsuffix])       Make a deep copy of this element and all its children of this element.         copy([newdoc, idsuffix])       Generator creating a deep copy of the children of this element.         count(Class[, set, recursive, ignore, node])       Like AbstractElement.select(), but instead of returning the elements, it merely counts them.         deepvalidation()       Perform deep validation of this element.         description()       Obtain the description associated with the element.         feat(subset)       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textelase by looking in the underlying corrections where it is reu	accepts(Class[, raiseexceptions, parentinstance])	
added to the parent.  addidsuffix(idsuffix[, recursive])  Appends a suffix to this element's ID, and optionally to all child IDs as well.  Addtoindex([norecurse])  Makes sure this element (and all subelements), are properly added to the index.  Find the most immediate ancestor of the specified type, multiple classes may be specified.  Ancestors([Class])  Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.  Append(child, *args, **kwargs)  context(size[, placeholder, scope])  Returns this word in context, {size} words to the left the current word, and {size} words to the right  copy([newdoc, idsuffix])  Cenerator creating a deep copy of this element and all its children copychildren([newdoc, idsuffix])  Cenerator creating a deep copy of the children of this element.  count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but in stead of returning the elements, it merely counts them.  deepvalidation()  Derform deep validation of this element.  description()  Obtain the description associated with the element.  feat(subset)  findcorrectionhandling(cls)  Find the proper correctionhandling given a textelase by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements	add(child, *args, **kwargs)	
addidsuffix(idsuffix[, recursive])       Appends a suffix to this element's ID, and optionally to all child IDs as well.         addtoindex([norecurse])       Makes sure this element (and all subelements), are properly added to the index.         ancestor(*Classes)       Find the most immediate ancestor of the specified type, multiple classes may be specified.         ancestors([Class])       Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.         append(child, *args, **kwargs)       Returns this word in context, {size} words to the left the current word, and {size} words to the right         context(size[, placeholder, scope])       Returns this word in context, {size} words to the right         copy([newdoc, idsuffix])       Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])         Generator creating a deep copy of the children of this element.       Count(Class[, set, recursive, ignore, node])       Like AbstractElement.select(), but in stead of returning the elements, it merely counts them.         deepvalidation()       Perform deep validation of this element.         description()       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at whi	addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
to all child IDs as well.  addtoindex([norecurse])  Makes sure this element (and all subelements), are properly added to the index.  ancestor(*Classes)  Find the most immediate ancestor of the specified type, multiple classes may be specified.  ancestors([Class])  Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.  append(child, *args, **kwargs)  context(size[, placeholder, scope])  Returns this word in context, {size} words to the left the current word, and {size} words to the right copy([newdoc, idsuffix])  Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])  Generator creating a deep copy of the children of this element.  count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but imstead of returning the elements, it merely counts them.  deepvalidation()  description()  Obtain the description associated with the element.  feat(subset)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements		added to the parent.
addtoindex([norecurse])         ancestor(*Classes)       Find the most immediate ancestor of the specified type, multiple classes may be specified.         ancestors([Class])       Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.         append(child, *args, **kwargs)       Returns this word in context, {size} words to the left the current word, and {size} words to the right         context(size[, placeholder, scope])       Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])         Generator creating a deep copy of the children of this element.       Count(Class[, set, recursive, ignore, node])         Like AbstractElement.select(), but instead of returning the elements, it merely counts them.       Deform deep validation of this element.         deepvalidation()       Perform deep validation of this element.         description()       Obtain the description associated with the element.         feat(subset)       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         get the index at which an element occurs, recursive by default!         get metadata ([key])       Get the metadata that applies to this element, automatically inherited from parent elements	addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
properly added to the index.  ancestor(*Classes)  Find the most immediate ancestor of the specified type, multiple classes may be specified.  ancestors([Class])  Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.  append(child, *args, **kwargs)  context(size[, placeholder, scope])  Returns this word in context, {size} words to the left the current word, and {size} words to the right  copy([newdoc, idsuffix])  Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])  Generator creating a deep copy of the children of this element.  count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  deepvalidation()  description()  Obtain the description associated with the element.  feat(subset)  findcorrectionhandling(cls)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  Internal method to find replaceable elements.  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements		
### ancestor (*Classes)  ### Find the most immediate ancestor of the specified type, multiple classes may be specified.  #### Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.  ########## Append(child, *args, **kwargs)  ###################################	addtoindex([norecurse])	Makes sure this element (and all subelements), are
type, multiple classes may be specified.  ancestors([Class])  Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.  append(child, *args, **kwargs)  context(size[, placeholder, scope])  Returns this word in context, {size} words to the left the current word, and {size} words to the right  copy([newdoc, idsuffix])  Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])  Generator creating a deep copy of the children of this element.  count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  deepvalidation()  description()  feat(subset)  findcorrectionhandling(cls)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements		properly added to the index.
ancestors([Class])         Generator yielding all ancestors of this element, effectively back-tracing its path to the root element.           append(child, *args, **kwargs)         Returns this word in context, {size} words to the left the current word, and {size} words to the right           copy([newdoc, idsuffix])         Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])           Generator creating a deep copy of the children of this element.           count(Class[, set, recursive, ignore, node])         Like AbstractElement.select(), but instead of returning the elements, it merely counts them.           deepvalidation()         Perform deep validation of this element.           description()         Obtain the description associated with the element.           feat(subset)         Obtain the feature class value of the specific subset.           findcorrectionhandling(cls)         Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused           findreplaceables(parent[, set])         Internal method to find replaceable elements.           getindex(child[, recursive, ignore])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements	ancestor(*Classes)	Find the most immediate ancestor of the specified
fectively back-tracing its path to the root element.  append(child, *args, **kwargs)  context(size[, placeholder, scope])  Returns this word in context, {size} words to the left the current word, and {size} words to the right  copy([newdoc, idsuffix])  Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])  Count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  deepvalidation()  Perform deep validation of this element.  description()  Obtain the description associated with the element.  feat(subset)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  Jinternal method to find replaceable elements.  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements		type, multiple classes may be specified.
append(child, *args, **kwargs)         context(size[, placeholder, scope])       Returns this word in context, {size} words to the left the current word, and {size} words to the right         copy([newdoc, idsuffix])       Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])         Generator creating a deep copy of the children of this element.         count(Class[, set, recursive, ignore, node])       Like AbstractElement.select(), but instead of returning the elements, it merely counts them.         deepvalidation()       Perform deep validation of this element.         description()       Obtain the description associated with the element.         feat(subset)       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements	ancestors([Class])	Generator yielding all ancestors of this element, ef-
context(size[, placeholder, scope])Returns this word in context, {size} words to the left the current word, and {size} words to the rightcopy([newdoc, idsuffix])Make a deep copy of this element and all its children copychildren([newdoc, idsuffix])count(Class[, set, recursive, ignore, node])Like AbstractElement.select(), but instead of returning the elements, it merely counts them.deepvalidation()Perform deep validation of this element.description()Obtain the description associated with the element.feat(subset)Obtain the feature class value of the specific subset.findcorrectionhandling(cls)Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reusedfindreplaceables(parent[, set])Internal method to find replaceable elements.getindex(child[, recursive, ignore])Get the index at which an element occurs, recursive by default!getmetadata([key])Get the metadata that applies to this element, automatically inherited from parent elements		fectively back-tracing its path to the root element.
the current word, and {size} words to the right  copy([newdoc, idsuffix])  Make a deep copy of this element and all its children  Generator creating a deep copy of the children of this element.  Count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  deepvalidation()  description()  Detain the description associated with the element.  feat(subset)  Cobtain the feature class value of the specific subset.  findcorrectionhandling(cls)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements		
copy([newdoc, idsuffix])       Make a deep copy of this element and all its children         copychildren([newdoc, idsuffix])       Generator creating a deep copy of the children of this element.         count(Class[, set, recursive, ignore, node])       Like AbstractElement.select(), but instead of returning the elements, it merely counts them.         deepvalidation()       Perform deep validation of this element.         description()       Obtain the description associated with the element.         feat(subset)       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textelast by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements	context(size[, placeholder, scope])	Returns this word in context, {size} words to the left,
copychildren([newdoc, idsuffix])       Generator creating a deep copy of the children of this element.         count(Class[, set, recursive, ignore, node])       Like AbstractElement.select(), but instead of returning the elements, it merely counts them.         deepvalidation()       Perform deep validation of this element.         description()       Obtain the description associated with the element.         feat(subset)       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements		
element.  Count(Class[, set, recursive, ignore, node])  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  deepvalidation()  Perform deep validation of this element.  description()  Obtain the description associated with the element.  feat(subset)  Obtain the feature class value of the specific subset.  findcorrectionhandling(cls)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements	copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
count(Class[, set, recursive, ignore, node])       Like AbstractElement.select(), but instead of returning the elements, it merely counts them.         deepvalidation()       Perform deep validation of this element.         description()       Obtain the description associated with the element.         feat(subset)       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements	copychildren([newdoc, idsuffix])	Generator creating a deep copy of the children of this
stead of returning the elements, it merely counts them.  deepvalidation()  Perform deep validation of this element.  description()  Obtain the description associated with the element.  feat(subset)  Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  Jinternal method to find replaceable elements.  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements		element.
them.  deepvalidation()  Perform deep validation of this element.  description()  Obtain the description associated with the element.  feat(subset)  Obtain the feature class value of the specific subset.  findcorrectionhandling(cls)  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements	count(Class[, set, recursive, ignore, node])	Like AbstractElement.select(), but in-
deepvalidation()         Perform deep validation of this element.           description()         Obtain the description associated with the element.           feat(subset)         Obtain the feature class value of the specific subset.           findcorrectionhandling(cls)         Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused           findreplaceables(parent[, set])         Internal method to find replaceable elements.           getindex(child[, recursive, ignore])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements		stead of returning the elements, it merely counts
description()         Obtain the description associated with the element.           feat(subset)         Obtain the feature class value of the specific subset.           findcorrectionhandling(cls)         Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused           findreplaceables(parent[, set])         Internal method to find replaceable elements.           getindex(child[, recursive, ignore])         Get the index at which an element occurs, recursive by default!           getmetadata([key])         Get the metadata that applies to this element, automatically inherited from parent elements		them.
feat(subset)       Obtain the feature class value of the specific subset.         findcorrectionhandling(cls)       Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements		
findcorrectionhandling(cls)       Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused         findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements	description()	
by looking in the underlying corrections where it is reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements		
reused  findreplaceables(parent[, set])  getindex(child[, recursive, ignore])  getmetadata([key])  Get the index at which an element occurs, recursive by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements	findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
findreplaceables(parent[, set])       Internal method to find replaceable elements.         getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements		by looking in the underlying corrections where it is
getindex(child[, recursive, ignore])       Get the index at which an element occurs, recursive by default!         getmetadata([key])       Get the metadata that applies to this element, automatically inherited from parent elements		
by default!  getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements	findreplaceables(parent[, set])	
getmetadata([key])  Get the metadata that applies to this element, automatically inherited from parent elements	<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
matically inherited from parent elements		by default!
	getmetadata([key])	Get the metadata that applies to this element, auto-
gettextdelimiter([retaintokenisation]) Return the text delimiter for this class.		
	<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])  Does this element have phonetic content (of the spec-	hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
ified class)		ified class)
hastext([cls, strict, correctionhandling])  Does this element have text (of the specified class)	hastext([cls, strict, correctionhandling])	
incorrection() Is this element part of a correction? If it is, it returns	incorrection()	Is this element part of a correction? If it is, it returns
		the Correction element (evaluating to True), other-
wise it returns None		wise it returns None
Continued on next page		Continued on next page

Table 103 – continued from previous page

	ued from previous page
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of <i>all</i> items below this element (not limited to AbstractElement)
json([attribs, recurse, ignorelist])	Serialises the FoLiA element and all its contents to a
JSOM([attribs, rectirse, ignorenst])	Python dictionary suitable for serialisation to JSON.
1.5+ cost cost(size[ placeholder seems])	Returns the left context for an element, as a list.
leftcontext(size[, placeholder, scope])	
next([Class, scope, reverse])	Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined
	scope.
originaltext([cls])	Alias for retrieving the original uncorrect text.
parsexml(node, doc, **kwargs)	Internal class method used for turning an XML element into an instance of the Class.
phon([cls, previousdelimiter, strict,])	Get the phonetic representation associated with this element (of the specified class)
phoncontent([cls, correctionhandling])	Get the phonetic content explicitly associated with
phoneontent([cis, correctionnaliding])	
1/\	this element (of the specified class).
postappend()	This method will be called after an element is added
	to another and does some checks.
previous([Class, scope])	Returns the previous element, if it is of the specified
	type and if it does not cross the boundary of the defined scope.
relaxng([includechildren, extraattribs,])	Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)
remove(child)	Removes the child element
replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace (cinia, args, kwargs)	places any existing child element of the same type
	and set.
resolve([alignmentcontext, documents])	and bot.
resolveword(id)	
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the scop.
settext(text[, cls])	Set the text for this element.
speech_speaker()	Retrieves the speaker of the audio or video file asso-
pheecii_pheaver()	ciated with the element.
anaah ana()	Retrieves the URL/filename of the audio or video file
speech_src()	
(5.1.3)	associated with the element.
stricttext([cls])	Alias for text() with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
-	Alias for text() with
LOKLEXL([CIS])	• • • • • • • • • • • • • • • • • • • •
LOKLEXL([CIS])	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of the children.

Table 103 – continued from previous page

xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
	XML.
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to
	XML.
iter()	Iterate over all children of this element.
len()	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text()

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo
ANNOTATIONTYPE = None
AUTH = True
AUTO_GENERATE_ID = False
OCCURRENCES = 0
OCCURRENCES_PER_SET = 0
OPTIONAL_ATTRIBS = None
PHONCONTAINER = False
PRIMARYELEMENT = True
PRINTABLE = False
REQUIRED_ATTRIBS = None
REQUIRED_DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'aref'
Method Details
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
__init__ (doc, *args, **kwargs)
    Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
```

### classmethod addable (parent, set=None, raiseexceptions=True)

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

### **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

#### Returns bool

Raises ValueError

## addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy ()

### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

### ancestor(\*Classes)

Find the most immediate ancestor of the specified type, multiple classes may be specified.

Parameters \*Classes - The possible classes (AbstractElement or subclasses) to select from. Not instances!

### Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

Yields elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

#### **Parameters**

- newdoc (Document) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

**Returns** a copy of the element

#### copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

## count (Class, set=None, recursive=True, ignore=True, node=None)

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

# findcorrection handling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

## classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

# getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

# gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

## **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon (), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

## incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

### originaltext (cls='original')

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

- node XML Element (\*) -
- doc Document (\*)-

**Returns** An instance of the current Class.

**phon** (cls='current', previousdelimiter=", strict=False, correctionhandling=1)

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon()*. Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

# Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

# phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

classmethod relaxng (includechildren=True, extraattribs=None, extraelements=None)

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

### remove (child)

Removes the child element

```
replace (child, *args, **kwargs)
```

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

# **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to) -

See AbstractElement.append() for more information and all parameters.

```
resolve (alignmentcontext=None, documents={})
```

```
resolveword(id)
```

```
rightcontext (size, placeholder=None, scope=None)
```

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
select (Class, set=None, recursive=True, ignore=True, node=None)
```

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

### Example:

# setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

# setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

# setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by <code>copy()</code>

```
settext (text, cls='current')
```

Set the text for this element.

### **Parameters**

• **text** (str) – The text

• **cls** (str) – The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize\_spaces=False)

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *text* (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

# Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

```
textcontent (cls='current', correctionhandling=1)
```

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (TextContent)

Raises NoSuchText if there is no text content for the element

### See also:

```
text() phoncontent() phon()
```

### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

Returns bool

```
toktext (cls='current')
```

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

## See also:

```
AbstractElement.xmlstring() - for direct string output
```

# xmlstring(pretty\_print=False)

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

### Return type str

```
__iter__()
```

Iterate over all children of this element.

Example:

```
for annotation in word:

__len__()
Returns the number of child elements under the current element.

__str__()
Alias for text()
```

# 4.4.6 Descriptions, Metrics

FoLiA allows arbitrary descriptions to be assigned with any element. It also allows assigning metrics to any annotation, which consist of a key/value pair that often express a quantivative or qualitative measure. This is accomplished, respectively, with the following element classes:

Description	Description is an element that can be used to associate a
	description with almost any other FoLiA element
Metric	Metric elements provide a key/value pair to allow the
	annotation of any kind of metric with any kind of anno-
	tation element.

# pynlpl.formats.folia.Description

```
class pynlpl.formats.folia.Description(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Description is an element that can be used to associate a description with almost any other FoLiA element

# **Method Summary**

init(doc, *args, **kwargs)	Required keyword arguments: * value=: The text
	content for the description (str or unicode)
<pre>accepts(Class[, raiseexceptions, parentinstance])</pre>	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
	Continued on next page

Table 105 –	<ul> <li>continued fron</li> </ul>	n previous page

Generator creating a deep copy of the children of this element.  Like AbstractElement.select(), but instead of returning the elements, it merely counts them.  Perform deep validation of this element.  Obtain the description associated with the element.  Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this element (not limited to AbstractElement)
stead of returning the elements, it merely counts them.  Perform deep validation of this element.  Obtain the description associated with the element.  Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None
stead of returning the elements, it merely counts them.  Perform deep validation of this element.  Obtain the description associated with the element.  Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None
them.  Perform deep validation of this element.  Obtain the description associated with the element.  Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this
Obtain the description associated with the element.  Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this
Obtain the description associated with the element.  Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this
Obtain the feature class value of the specific subset.  Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None
Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this
by looking in the underlying corrections where it is reused  Internal method to find replaceable elements.  Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this
Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this
Get the index at which an element occurs, recursive by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of all items below this
by default!  Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
Get the metadata that applies to this element, automatically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
matically inherited from parent elements  Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
Return the text delimiter for this class.  Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
Does this element have phonetic content (of the specified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
ified class)  Does this element have text (of the specified class)  Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
Does this element have text (of the specified class) Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
the Correction element (evaluating to True), otherwise it returns None  Returns a depth-first flat list of <i>all</i> items below this
wise it returns None  Returns a depth-first flat list of <i>all</i> items below this
Returns a depth-first flat list of <i>all</i> items below this
element (not limited to AbstractElement)
Serialises the FoLiA element and all its contents to a
Python dictionary suitable for serialisation to JSON.
Returns the left context for an element, as a list.
Returns the next element, if it is of the specified type
and if it does not cross the boundary of the defined
scope.
Alias for retrieving the original uncorrect text.
Internal class method used for turning an XML ele-
ment into an instance of the Class.
Get the phonetic representation associated with this
element (of the specified class)
Get the phonetic content explicitly associated with
this element (of the specified class).
This method will be called after an element is added
to another and does some checks.
Returns the previous element, if it is of the specified
type and if it does not cross the boundary of the de-
fined scope.
Returns a RelaxNG definition for this element (as an
XML element (lxml.etree) rather than a string)
Removes the child element
Appends a child element like append(), but re-
places any existing child element of the same type
and set.

Continued on next page

Table 105 – continued from previous page

rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.	
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.	
setdoc(newdoc)	Set a different document.	
setdocument(doc)	Associate a document with this element.	
setparents()	Correct all parent relations for elements within the	
	scop.	
settext(text[, cls])	Set the text for this element.	
speech_speaker()	Retrieves the speaker of the audio or video file asso-	
	ciated with the element.	
speech_src()	Retrieves the URL/filename of the audio or video file	
	associated with the element.	
stricttext([cls])	Alias for text() with strict=True	
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-	
, ,	ified class)	
textcontent([cls, correctionhandling])	Get the text content explicitly associated with this	
(2)	element (of the specified class).	
textvalidation([warnonly])	Run text validation on this element.	
toktext([cls])	Alias for text() with	
	retaintokenisation=True	
updatetext()	Recompute textual value based on the text content of	
	the children.	
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to	
([utures, erements, simpermeren])	XML.	
xmlstring([pretty_print])	Serialises this FoLiA element and all its contents to	
xmisering ([piotty_pimt])	XML.	
iter()	Iterate over all children of this element.	
len ()	Returns the number of child elements under the cur-	
	rent element.	
str ()	Alias for text()	
	THIRD TOT COAC ( )	

## **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Description'>, <class 'pynlpl.formats.fo

ANNOTATIONTYPE = None

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Description'

OCCURRENCES = 1

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 2, 3, 5, 4, 11)

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = False

REQUIRED_ATTRIBS = None

REQUIRED_DATA = None
```

```
SETONLY = False

SPEAKABLE = False

SUBSET = None

TEXTCONTAINER = False

TEXTDELIMITER = None

XLINK = False

XMLTAG = 'desc'
```

### **Method Details**

```
__init__ (doc, *args, **kwargs)
    Required keyword arguments: * value=: The text content for the description (str or unicode)
__init__ (doc, *args, **kwargs)
    Required keyword arguments: * value=: The text content for the description (str or unicode)

classmethod accepts (Class, raiseexceptions=True, parentinstance=None)

add (child, *args, **kwargs)
```

 $\verb|classmethod| addable| (parent, set=None, raise exceptions=True)$ 

Tests whether a new element of this class can be added to the parent.

This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden by subclasses for more customised behaviour.

# **Parameters**

- parent (AbstractElement) The element that is being added to
- set (str or None) The set
- raiseexceptions (bool) Raise an exception if the element can't be added?

### Returns bool

Raises ValueError

# addidsuffix (idsuffix, recursive=True)

Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to call this directly, invoked implicitly by copy()

#### addtoindex (norecurse=[])

Makes sure this element (and all subelements), are properly added to the index.

Mostly for internal use.

```
ancestor(*Classes)
```

Find the most immediate ancestor of the specified type, multiple classes may be specified.

**Parameters** \*Classes – The possible classes (AbstractElement or subclasses) to select from. Not instances!

# Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

**Parameters** \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right

```
copy (newdoc=None, idsuffix=")
```

Make a deep copy of this element and all its children.

### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

## copychildren (newdoc=None, idsuffix=")

Generator creating a deep copy of the children of this element.

Invokes *copy* () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

### Returns int

# deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

### description()

Obtain the description associated with the element.

**Raises** NoSuchAnnotation if there is no associated description.

### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

### Returns str or list

## findcorrectionhandling(cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

#### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by AbstractElement. replace(). Can be overriden for more fine-grained control.

## getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata(key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

## gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

# **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Returns bool

hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text (), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

#### items (founditems=[])

Returns a depth-first flat list of all items below this element (not limited to AbstractElement)

# json (attribs=None, recurse=True, ignorelist=False)

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

# leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may
  also be a tuple of multiple classes. Set to True to constrain to the same class as that of
  the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext(cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

# classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### **Parameters**

```
node - XML Element (*) -doc - Document (*) -
```

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- retaintokenisation (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.

- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

## phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

## **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (PhonContent)

Raises NoSuchPhon if there is no phonetic content for the element

### See also:

```
phon() textcontent() text()
```

### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

```
previous (Class=True, scope=True)
```

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative.  $(t \circ)$  -

See AbstractElement.append() for more information and all parameters.

resolveword(id)

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

select (Class, set=None, recursive=True, ignore=True, node=None)

Select child elements of the specified class.

A further restriction can be made based on set.

### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (*str*) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- recursive (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

Example:

### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument (doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by copy()

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (str) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

**Returns** str or None if not found

# speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
text (cls='current', retaintokenisation=False, previousdelimiter=", strict=False, correctionhan-dling=1, normalize_spaces=False)
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

# **Parameters**

- cls (str) The class of the text content to obtain, defaults to current.
- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.

- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text (). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.
- normalize\_spaces (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (*str*) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

# textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

# Returns bool

# toktext (cls='current')

Alias for text () with retaintokenisation=True

### updatetext()

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

**xml** (attribs=None, elements=None, skipchildren=False)

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

AbstractElement.xmlstring() - for direct string output

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
___iter___()
```

Iterate over all children of this element.

Example:

```
for annotation in word:
    ...
```

\_\_len\_\_()

Returns the number of child elements under the current element.

```
___str__()
Alias for text()
```

# pynlpl.formats.folia.Metric

```
class pynlpl.formats.folia.Metric(doc, *args, **kwargs)
    Bases: pynlpl.formats.folia.AbstractElement
```

Metric elements provide a key/value pair to allow the annotation of any kind of metric with any kind of annotation element.

It is used for example for statistical measures to be added to elements as annotation.

# **Method Summary**

init(doc, *args, **kwargs)	Initialize self.
accepts(Class[, raiseexceptions, parentinstance])	
add(child, *args, **kwargs)	
addable(parent[, set, raiseexceptions])	Tests whether a new element of this class can be
	added to the parent.
addidsuffix(idsuffix[, recursive])	Appends a suffix to this element's ID, and optionally
	to all child IDs as well.
addtoindex([norecurse])	Makes sure this element (and all subelements), are
	properly added to the index.
ancestor(*Classes)	Find the most immediate ancestor of the specified
	type, multiple classes may be specified.
ancestors([Class])	Generator yielding all ancestors of this element, ef-
	fectively back-tracing its path to the root element.
	Continued on next page

Chapter 4. FoLiA library

Table 106 – continued from previous page

ann an d'abild *anna **!anac'	ued from previous page
append(child, *args, **kwargs)	
<pre>context(size[, placeholder, scope])</pre>	Returns this word in context, {size} words to the left,
	the current word, and {size} words to the right
copy([newdoc, idsuffix])	Make a deep copy of this element and all its children.
<pre>copychildren([newdoc, idsuffix])</pre>	Generator creating a deep copy of the children of this
	element.
<pre>count(Class[, set, recursive, ignore, node])</pre>	Like AbstractElement.select(), but in-
	stead of returning the elements, it merely counts
	them.
deepvalidation()	Perform deep validation of this element.
description()	Obtain the description associated with the element.
feat(subset)	Obtain the feature class value of the specific subset.
findcorrectionhandling(cls)	Find the proper correctionhandling given a textclass
Timacottecetomanating(cis)	by looking in the underlying corrections where it is
	reused
findran la cach la c(narant[ cat])	Internal method to find replaceable elements.
findreplaceables(parent[, set])	
<pre>getindex(child[, recursive, ignore])</pre>	Get the index at which an element occurs, recursive
([1 ])	by default!
getmetadata([key])	Get the metadata that applies to this element, auto-
	matically inherited from parent elements
<pre>gettextdelimiter([retaintokenisation])</pre>	Return the text delimiter for this class.
hasphon([cls, strict, correctionhandling])	Does this element have phonetic content (of the spec-
	ified class)
hastext([cls, strict, correctionhandling])	Does this element have text (of the specified class)
incorrection()	Is this element part of a correction? If it is, it returns
	the Correction element (evaluating to True), other-
	wise it returns None
<pre>insert(index, child, *args, **kwargs)</pre>	
<pre>items([founditems])</pre>	Returns a depth-first flat list of all items below this
	element (not limited to AbstractElement)
<pre>json([attribs, recurse, ignorelist])</pre>	Serialises the FoLiA element and all its contents to a
	Python dictionary suitable for serialisation to JSON.
<pre>leftcontext(size[, placeholder, scope])</pre>	Returns the left context for an element, as a list.
next([Class, scope, reverse])	Returns the next element, if it is of the specified type
nexe([elass, scope, reverse])	and if it does not cross the boundary of the defined
	-
original+ov+([cls])	scope.
originaltext([cls])	scope.  Alias for retrieving the original uncorrect text.
<pre>originaltext([cls]) parsexml(node, doc, **kwargs)</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML ele-
<pre>parsexml(node, doc, **kwargs)</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.
	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)
parsexml(node, doc, **kwargs)	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])  postappend()</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])  postappend()</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the de-
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])  postappend()</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.  Returns a RelaxNG definition for this element (as an
<pre>parsexml(node, doc, **kwargs)  phon([cls, previousdelimiter, strict,])  phoncontent([cls, correctionhandling])  postappend()  previous([Class, scope])</pre>	scope.  Alias for retrieving the original uncorrect text.  Internal class method used for turning an XML element into an instance of the Class.  Get the phonetic representation associated with this element (of the specified class)  Get the phonetic content explicitly associated with this element (of the specified class).  This method will be called after an element is added to another and does some checks.  Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope.

Table 106 – continued from previous page

replace(child, *args, **kwargs)	Appends a child element like append(), but re-
reprace (cinia, args, kwargs)	places any existing child element of the same type
	and set.
resolveword(id)	und set.
rightcontext(size[, placeholder, scope])	Returns the right context for an element, as a list.
select(Class[, set, recursive, ignore, node])	Select child elements of the specified class.
setdoc(newdoc)	Set a different document.
setdocument(doc)	Associate a document with this element.
setparents()	Correct all parent relations for elements within the
setpatents()	_
	Set the text for this element.
settext(text[, cls])	
speech_speaker()	Retrieves the speaker of the audio or video file asso-
, ,	ciated with the element.
speech_src()	Retrieves the URL/filename of the audio or video file
	associated with the element.
stricttext([cls])	Alias for text () with strict=True
text([cls, retaintokenisation,])	Get the text associated with this element (of the spec-
	ified class)
<pre>textcontent([cls, correctionhandling])</pre>	Get the text content explicitly associated with this
	element (of the specified class).
textvalidation([warnonly])	Run text validation on this element.
toktext([cls])	Alias for text() with
	retaintokenisation=True
updatetext()	Recompute textual value based on the text content of
	the children.
xm1([attribs, elements, skipchildren])	Serialises the FoLiA element and all its contents to
-	XML.
<pre>xmlstring([pretty_print])</pre>	Serialises this FoLiA element and all its contents to
5 (A	XML.
iter()	Iterate over all children of this element.
	Returns the number of child elements under the cur-
	rent element.
str()	Alias for text ()
	**

#### **Class Attributes**

```
ACCEPTED_DATA = (<class 'pynlpl.formats.folia.Comment'>, <class 'pynlpl.formats.folia.

ANNOTATIONTYPE = 30

AUTH = True

AUTO_GENERATE_ID = False

LABEL = 'Metric'

OCCURRENCES = 0

OCCURRENCES_PER_SET = 0

OPTIONAL_ATTRIBS = (0, 1, 2, 4, 3, 5, 8, 6, 7, 9, 11)

PHONCONTAINER = False

PRIMARYELEMENT = True

PRINTABLE = False
```

```
REQUIRED ATTRIBS = None
REQUIRED DATA = None
SETONLY = False
SPEAKABLE = False
SUBSET = None
TEXTCONTAINER = False
TEXTDELIMITER = None
XLINK = False
XMLTAG = 'metric'
Method Details
init (doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
__init__(doc, *args, **kwargs)
     Initialize self. See help(type(self)) for accurate signature.
classmethod accepts (Class, raiseexceptions=True, parentinstance=None)
add (child, *args, **kwargs)
classmethod addable (parent, set=None, raiseexceptions=True)
     Tests whether a new element of this class can be added to the parent.
     This method is mostly for internal use. This will use the OCCURRENCES property, but may be overidden
     by subclasses for more customised behaviour.
         Parameters
             • parent (AbstractElement) - The element that is being added to
             • set (str or None) - The set
             • raiseexceptions (bool) – Raise an exception if the element can't be added?
         Returns bool
         Raises ValueError
addidsuffix (idsuffix, recursive=True)
     Appends a suffix to this element's ID, and optionally to all child IDs as well. There is sually no need to
     call this directly, invoked implicitly by copy ()
addtoindex (norecurse=[])
     Makes sure this element (and all subelements), are properly added to the index.
     Mostly for internal use.
ancestor(*Classes)
     Find the most immediate ancestor of the specified type, multiple classes may be specified.
         Parameters *Classes - The possible classes (AbstractElement or subclasses) to select
             from. Not instances!
```

Example:

```
paragraph = word.ancestor(folia.Paragraph)
```

#### ancestors(Class=None)

Generator yielding all ancestors of this element, effectively back-tracing its path to the root element. A tuple of multiple classes may be specified.

Parameters \*Class - The class or classes (AbstractElement or subclasses). Not instances!

**Yields** elements (instances derived from AbstractElement)

```
append (child, *args, **kwargs)
```

```
context (size, placeholder=None, scope=None)
```

Returns this word in context, {size} words to the left, the current word, and {size} words to the right copy (newdoc=None, idsuffix=")

Make a deep copy of this element and all its children.

#### **Parameters**

- **newdoc** (*Document*) The document the copy should be associated with.
- idsuffix (str or bool) If set to a string, the ID of the copy will be append with this (prevents duplicate IDs when making copies for the same document). If set to True, a random suffix will be generated.

Returns a copy of the element

```
copychildren (newdoc=None, idsuffix=")
```

Generator creating a deep copy of the children of this element.

Invokes copy () on all children, parameters are the same.

```
count (Class, set=None, recursive=True, ignore=True, node=None)
```

Like AbstractElement.select(), but instead of returning the elements, it merely counts them.

#### Returns int

#### deepvalidation()

Perform deep validation of this element.

```
Raises DeepValidationError
```

#### description()

Obtain the description associated with the element.

Raises NoSuchAnnotation if there is no associated description.

#### feat (subset)

Obtain the feature class value of the specific subset.

If a feature occurs multiple times, the values will be returned in a list.

#### Example:

```
sense = word.annotation(folia.Sense)
synset = sense.feat('synset')
```

#### Returns str or list

#### findcorrectionhandling (cls)

Find the proper correctionhandling given a textclass by looking in the underlying corrections where it is reused

#### classmethod findreplaceables (parent, set=None, \*\*kwargs)

Internal method to find replaceable elements. Auxiliary function used by <code>AbstractElement.replace()</code>. Can be overriden for more fine-grained control.

#### getindex (child, recursive=True, ignore=True)

Get the index at which an element occurs, recursive by default!

#### Returns int

#### getmetadata (key=None)

Get the metadata that applies to this element, automatically inherited from parent elements

#### gettextdelimiter (retaintokenisation=False)

Return the text delimiter for this class.

Uses the TEXTDELIMITER attribute but may return a customised one instead.

#### **hasphon** (*cls='current'*, *strict=True*, *correctionhandling=1*)

Does this element have phonetic content (of the specified class)

By default, and unlike phon(), this checks strictly, i.e. the element itself must have the phonetic content and it is not inherited from its children.

#### **Parameters**

- **cls** (*str*) The class of the phonetic content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what phonetic content to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### hastext (cls='current', strict=True, correctionhandling=1)

Does this element have text (of the specified class)

By default, and unlike text(), this checks strictly, i.e. the element itself must have the text and it is not inherited from its children.

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- **strict** (bool) Set this if you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to True.
- correctionhandling Specifies what text to check for when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Returns bool

#### incorrection()

Is this element part of a correction? If it is, it returns the Correction element (evaluating to True), otherwise it returns None

```
insert (index, child, *args, **kwargs)
```

```
items (founditems=[])
```

Returns a depth-first flat list of *all* items below this element (not limited to AbstractElement)

```
json (attribs=None, recurse=True, ignorelist=False)
```

Serialises the FoLiA element and all its contents to a Python dictionary suitable for serialisation to JSON.

#### Example:

```
import json
json.dumps(word.json())
```

#### Returns dict

#### leftcontext (size, placeholder=None, scope=None)

Returns the left context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

```
next (Class=True, scope=True, reverse=False)
```

Returns the next element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement', may also be a tuple of multiple classes. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

```
originaltext (cls='original')
```

Alias for retrieving the original uncorrect text.

A call to text () with correctionhandling=CorrectionHandling.ORIGINAL

#### classmethod parsexml (node, doc, \*\*kwargs)

Internal class method used for turning an XML element into an instance of the Class.

#### Parameters

```
• node - XML Element (*) -
```

• doc - Document (\*)-

**Returns** An instance of the current Class.

```
phon (cls='current', previousdelimiter=", strict=False, correctionhandling=1)
```

Get the phonetic representation associated with this element (of the specified class)

The phonetic content will be constructed from child-elements whereever possible, as they are more specific. If no phonetic content can be obtained from the children and the element has itself phonetic content associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the phonetic content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and phonetic content will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (*str*) Can be set to a delimiter that was last outputed, useful when chaining calls to *phon* (). Defaults to an empty string.
- **strict** (bool) Set this if you are strictly interested in the phonetic content explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what phonetic content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current phonetic content. You can set this to CorrectionHandling.ORIGINAL if you want the phonetic content prior to correction, and CorrectionHandling.EITHER if you don't care.

#### Example:

```
word.phon()
```

**Returns** The phonetic content of the element (unicode instance in Python 2, str in Python 3)

**Raises** NoSuchPhon – if no phonetic conent is found at all.

#### See also:

phoncontent(): Retrieves the phonetic content as an element rather than a string text()
textcontent()

#### phoncontent (cls='current', correctionhandling=1)

Get the phonetic content explicitly associated with this element (of the specified class).

Unlike *phon()*, this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the PhonContent instance rather than the actual text!

#### **Parameters**

- cls (str) The class of the phonetic content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*PhonContent*)

Raises NoSuchPhon if there is no phonetic content for the element

#### See also:

```
phon() textcontent() text()
```

#### postappend()

This method will be called after an element is added to another and does some checks.

It can do extra checks and if necessary raise exceptions to prevent addition. By default makes sure the right document is associated.

This method is mostly for internal use.

#### previous (Class=True, scope=True)

Returns the previous element, if it is of the specified type and if it does not cross the boundary of the defined scope. Returns None if no next element is found. Non-authoritative elements are never returned.

#### **Parameters**

- Class (\*) The class to select; any python class subclassed off 'AbstractElement'. Set to True to constrain to the same class as that of the current instance, set to None to not constrain at all
- **scope** (\*) A list of classes which are never crossed looking for a next element. Set to True to constrain to a default list of structure elements (Sentence, Paragraph, Division, Event, ListItem, Caption), set to None to not constrain at all.

Returns a RelaxNG definition for this element (as an XML element (lxml.etree) rather than a string)

remove (child)

Removes the child element

replace (child, \*args, \*\*kwargs)

Appends a child element like append(), but replaces any existing child element of the same type and set. If no such child element exists, this will act the same as append()

#### **Keyword Arguments**

- alternative (bool) If set to True, the *replaced* element will be made into an alternative. Simply use *AbstractElement.append()* if you want the added element
- be an alternative. (to)-

See AbstractElement.append() for more information and all parameters.

 $\verb"resolveword"\,(id)$ 

rightcontext (size, placeholder=None, scope=None)

Returns the right context for an element, as a list. This method crosses sentence/paragraph boundaries by default, which can be restricted by setting scope

**select** (*Class*, *set=None*, *recursive=True*, *ignore=True*, *node=None*)

Select child elements of the specified class.

A further restriction can be made based on set.

#### **Parameters**

- Class (class) The class to select; any python class (not instance) subclassed off AbstractElement
- **Set** (str) The set to match against, only elements pertaining to this set will be returned. If set to None (default), all elements regardless of set will be returned.
- **recursive** (bool) Select recursively? Descending into child elements? Defaults to True.
- ignore A list of Classes to ignore, if set to True instead of a list, all non-authoritative elements will be skipped (this is the default behaviour and corresponds to the following elements: Alternative, AlternativeLayer, Suggestion, and folia. Original. These elements and those contained within are never authorative. You may also include the boolean True as a member of a list, if you want to skip additional tags along the predefined non-authoritative ones.
- **node** (\*) Reserved for internal usage, used in recursion.

**Yields** Elements (instances derived from AbstractElement)

#### Example:

#### setdoc(newdoc)

Set a different document. Usually no need to call this directly, invoked implicitly by copy ()

#### setdocument(doc)

Associate a document with this element.

```
Parameters doc (Document) - A document
```

Each element must be associated with a FoLiA document.

#### setparents()

Correct all parent relations for elements within the scop. There is sually no need to call this directly, invoked implicitly by *copy()* 

```
settext (text, cls='current')
```

Set the text for this element.

#### **Parameters**

- text (str) The text
- **cls** (*str*) The class of the text, defaults to current (leave this unless you know what you are doing). There may be only one text content element of each class associated with the element.

#### speech\_speaker()

Retrieves the speaker of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

#### speech\_src()

Retrieves the URL/filename of the audio or video file associated with the element.

The source is inherited from ancestor elements if none is specified. For this reason, always use this method rather than access the src attribute directly.

Returns str or None if not found

```
stricttext (cls='current')
Alias for text() with strict=True
```

```
\begin{tabular}{ll} \textbf{text} (cls='current', retaintokenisation=False, previous delimiter=", strict=False, correction handling=1, normalize\_spaces=False) \\ \end{tabular}
```

Get the text associated with this element (of the specified class)

The text will be constructed from child-elements whereever possible, as they are more specific. If no text can be obtained from the children and the element has itself text associated with it, then that will be used.

#### **Parameters**

• cls (str) - The class of the text content to obtain, defaults to current.

- **retaintokenisation** (bool) If set, the space attribute on words will be ignored, otherwise it will be adhered to and text will be detokenised as much as possible. Defaults to False.
- **previousdelimiter** (str) Can be set to a delimiter that was last outputed, useful when chaining calls to text(). Defaults to an empty string.
- **strict** (bool) Set this iif you are strictly interested in the text explicitly associated with the element, without recursing into children. Defaults to False.
- correctionhandling Specifies what text to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current text. You can set this to CorrectionHandling.ORIGINAL if you want the text prior to correction, and CorrectionHandling.EITHER if you don't care
- **normalize\_spaces** (bool) Return the text with multiple spaces, linebreaks, tabs normalized to single spaces

#### Example:

word.text()

**Returns** The text of the element (unicode instance in Python 2, str in Python 3)

Raises NoSuchText - if no text is found at all.

#### textcontent (cls='current', correctionhandling=1)

Get the text content explicitly associated with this element (of the specified class).

Unlike text(), this method does not recurse into child elements (with the sole exception of the Correction/New element), and it returns the TextContent instance rather than the actual text!

#### **Parameters**

- **cls** (str) The class of the text content to obtain, defaults to current.
- correctionhandling Specifies what content to retrieve when corrections are encountered. The default is CorrectionHandling.CURRENT, which will retrieve the corrected/current content. You can set this to CorrectionHandling.ORIGINAL if you want the content prior to correction, and CorrectionHandling.EITHER if you don't care.

**Returns** The phonetic content (*TextContent*)

Raises NoSuchText if there is no text content for the element

#### See also:

```
text() phoncontent() phon()
```

#### textvalidation(warnonly=None)

Run text validation on this element. Checks whether any text redundancy is consistent and whether offsets are valid.

**Parameters warnonly** (bool) – Warn only (True) or raise exceptions (False). If set to None then this value will be determined based on the document's FoLiA version (Warn only before FoLiA v1.5)

#### Returns bool

```
toktext (cls='current')
```

Alias for text () with retain token is at ion=True

```
updatetext()
```

Recompute textual value based on the text content of the children. Only supported on elements that are a TEXTCONTAINER

```
xml (attribs=None, elements=None, skipchildren=False)
```

Serialises the FoLiA element and all its contents to XML.

Arguments are mostly for internal use.

Returns an lxml.etree.Element

See also:

```
AbstractElement.xmlstring() - for direct string output
```

```
xmlstring(pretty_print=False)
```

Serialises this FoLiA element and all its contents to XML.

**Returns** a string with XML representation for this element and all its children

Return type str

```
__iter__()
```

Iterate over all children of this element.

Example:

Alias for text ()

```
for annotation in word:

...

__len__()

Returns the number of child elements under the current element.

__str__()
```

#### 4.5 Metadata

FoLiA can be used with a variety of more advanced metadata schemes (e.g. Dublin Core, CMDI). If this is too much, you can use its own simple *native* metadata facility, a simple key value store. After instantiation of a *Document*, the metadata can be accessed through the metadata attribute, which behaves like a Python dictionary:

```
doc = folia.Document(file="/path/to/document.xml")
doc.metadata['language'] = "en"
```

4.5. Metadata 1051

# CHAPTER 5

**Formats** 

## 5.1 Corpus Gesproken Nederlands

```
exception pynlpl.formats.cgn.InvalidFeatureException
exception pynlpl.formats.cgn.InvalidTagException
pynlpl.formats.cgn.parse_cgn_postag(rawtag, raisefeatureexceptions=False)
```

#### 5.2 FoLiA

See folia: folia.html

## 5.3 GIZA++

reset()

```
targetword (index, targetwords, alignment)
          Return the aligned targeword for a specified index in the source words. Multiple words are concatenated
          together with a space in between
     targetwords (index, targetwords, alignment)
          Return the aligned targetwords for a specified index in the source words
class pynlpl.formats.giza.WordAlignment(filename, encoding=False)
     Target to Source alignment: reads target-source. A3. final files, in which each source word is aligned to one target
     word
     reset()
     targetword (index, targetwords, alignment)
          Return the aligned targetword for a specified index in the source words
pynlpl.formats.giza.parseAlignment(tokens)
5.4 Moses
class pynlpl.formats.moses.PhraseTable (filename, quiet=False, reverse=False,
                                                   iter='|||',
                                                               score\ column=3,
                                                                                   max sourcen=0,
                                                   sourceencoder=None, targetencoder=None, score-
                                                   filter=None)
class pynlpl.formats.moses.PhraseTableClient (host='localhost', port=65432)
5.5 SoNaR
class pynlpl.formats.sonar.Corpus(corpusdir,
                                                         extension='pos',
                                                                           restrict_to_collection=",
                                            conditionf=<function
                                                                 Corpus.<lambda>>,
                                                                                         ignoreer-
                                             rors=False)
class pynlpl.formats.sonar.CorpusDocument (filename, encoding='iso-8859-15')
     This class represent one document/text of the Corpus (read-only)
     paragraphs (with id=False)
          Extracts paragraphs, returns list of plain-text(!) paragraphs
     sentences()
          Iterate over all sentences (sentence_id, sentence) in the document, sentence is a list of 4-tuples
          (word,id,pos,lemma)
     words()
class pynlpl.formats.sonar.CorpusDocumentX(filename, tree=None, index=True)
     This class represent one document/text of the Corpus, loaded into memory at once and retaining the full structure
     paragraphs (node=None)
          iterate over paragraphs
     save (filename=None, encoding='iso-8859-15')
     sentences (node=None)
          iterate over sentences
```

1054 Chapter 5. Formats

```
validate (formats_dir='../formats/')
          checks if the document is valid
     words (node=None)
          iterate over words
     xpath (expression)
          Executes an xpath expression using the correct namespaces
class pynlpl.formats.sonar.CorpusFiles(corpusdir,
                                                                     extension='pos',
                                                  strict_to_collection=",
                                                                             conditionf=<function
                                                  Corpus.<lambda>>, ignoreerrors=False)
class pynlpl.formats.sonar.CorpusX(corpusdir, extension='pos', restrict_to_collection=",
                                             conditionf=<function Corpus.<lambda>>, ignoreer-
                                             rors=False)
pynlpl.formats.sonar.ns (namespace)
     Resolves the namespace identifier to a full URL
```

## 5.6 Taggerdata

```
class pynlpl.formats.taggerdata.Taggerdata(filename, encoding='utf-8', mode='r')
    align(referencewords, datatuple)
        align the reference sentence with the tagged data
    close()
    next()
    reset()
    write(sentence)
```

## 5.7 TiMBL

5.6. Taggerdata 1055

1056 Chapter 5. Formats

## Language Models

```
encoder=None,
class pynlpl.lm.lm.ARPALanguageModel (filename,
                                                                 encoding='utf-8',
                                                  base e=True,
                                                                    dounknown=True,
                                                                                         debug=False,
                                                  mode='simple')
     Full back-off language model, loaded from file in ARPA format.
     This class does not build the model but allows you to use a pre-computed one. You can use the tool ngram-count
     from for instance SRILM to actually build the model.
     class NgramsProbs (data, mode='simple', delim=' ')
          Store Ngrams with their probabilities and backoffs.
          This class is used in order to abstract the physical storage layout, and enable memory/speed tradeoffs.
          backoff(ngram)
               Return backoff value of a given ngram tuple
          prob (ngram)
               Return probability of given ngram tuple
     score (data, history=None)
     scoreword (word, history=None)
class pynlpl.lm.lm.SimpleLanguageModel (n=2, casesensitive=True, beginmarker='<br/>begin>',
     endmarker='<end>')
This is a simple unsmoothed language model. This class can both hold and compute the model.
     append (sentence)
     load (filename)
     save (filename)
     scoresentence (sentence)
class pynlpl.lm.srilm.SRILM(filename, n)
     logscore (ngram)
```

```
scoresentence (sentence, unknownwordprob=-12)
exception pynlpl.lm.srilm.SRILMException
    Base Exception for SRILM.
class pynlpl.lm.client.LMClient (host='localhost', port=12346, n=0)
scoresentence (sentence)
```

## Search Algorithms

This module contains various search algorithms.

```
class pynlpl.search.AbstractSearch(**kwargs)
     prune (state)
           Pruning method is called AFTER expansion of each node
     reset()
     searchall()
           Returns a list of all solutions
     searchbest()
           Returns the single best result (if multiple have the same score, the first match is returned)
     searchfirst()
           Returns the very first result (regardless of it being the best or not!)
     searchlast(n=10)
           Return the last n results (or possibly less if not found). Note that the last results are not necessarily the best
           ones! Depending on the search type.
     searchtop(n=10)
           Return the top n best resulta (or possibly less if not enough is found)
     traversal()
```

Returns all visited states (only when keeptraversal=True), note that this is not equal to the path, but contains

Returns the number of nodes visited (also when keeptravel=False). Note that this is not equal to the path,

visited(state)
class pynlpl.search.AbstractSearchState(parent=None, cost=0)

but contains all states that were checked!

all states that were checked!

traversalsize()

```
depth()
     expand()
          Generates successor states, implement your custom operators in the derived method.
     path()
     pathcost()
     score()
          Should return a heuristic value. This needs to be set if you plan to used an informed search algorithm.
     test (goalstates=None)
          Checks whether this state is a valid goal state, returns a boolean. If no goalstate is defined, then all states
          will test positively, this is what you usually want for optimisation problems.
class pynlpl.search.BeamSearch(states, beamsize, **kwargs)
     Local beam search algorithm
class pynlpl.search.BeamedBestFirstSearch(states, beamsize, **kwargs)
     Best first search with a beamsize (non-optimal!)
     prune (state)
          Pruning method is called AFTER expansion of each node
class pynlpl.search.BestFirstSearch(state, **kwargs)
class pynlpl.search.BreadthFirstSearch(state, **kwargs)
class pynlpl.search.DepthFirstSearch(state, **kwargs)
class pynlpl.search.EarlyEagerBeamSearch (state, beamsize, **kwargs)
     A beam search that prunes early (after each state expansion) and eagerly (weeding out worse successors)
     prune (state)
          Pruning method is called AFTER expansion of each node
class pynlpl.search.HillClimbingSearch(state, **kwargs)
     (identical to beamsearch with beam 1, but implemented differently)
class pynlpl.search.IterativeDeepening(state, **kwargs)
     traversal()
          Returns all visited states (only when keeptraversal=True), note that this is not equal to the path, but contains
          all states that were checked!
     traversalsize()
          Returns the number of nodes visited (also when keeptravel=False). Note that this is not equal to the path,
          but contains all states that were checked!
class pynlpl.search.StochasticBeamSearch(states, beamsize, **kwargs)
     prune (state)
          Pruning method is called AFTER expansion of each node
pynlpl.search.binary_search(a, x, lo=0, hi=None)
```

## Statistics and Information Theory

This module contains classes and functions for statistics and information theory. It is imported as follows:

```
import pynlpl.statistics
```

## 8.1 Generic functions

Amongst others, the following generic statistical functions are available:

```
* ``mean(list)`` - Computes the mean of a given list of numbers
```

- median (list) Computes the median of a given list of numbers
- stddev(list) Computes the standard deviation of a given list of numbers
- normalize (list) Normalizes a list of numbers so that the sum is 1.0.

## 8.2 Frequency Lists and Distributions

One of the most basic and widespread tasks in NLP is the creation of a frequency list. Counting is established by simply appending lists to the frequencylist:

```
freqlist = pynlpl.statistics.FrequencyList()
freqlist.append(['to','be','or','not','to','be'])
```

Take care not to append lists rather than strings unless you mean to create a frequency list over its characters rather than words. You may want to use the pynlpl.textprocessors.crudetokeniser first:

```
freqlist.append(pynlpl.textprocessors.crude_tokeniser("to be or not to be"))
```

The count can also be incremented explicitly explicitly for a single item:

```
freqlist.count('shakespeare')
```

The FrequencyList offers dictionary-like access. For example, the following statement will be true for the frequency list just created:

```
freqlist['be'] == 2
```

Normalised counts (pseudo-probabilities) can be obtained using the p () method:

```
freqlist.p('be')
```

Normalised counts can also be obtained by instantiation a Distribution instance using the frequency list:

```
dist = pynlpl.statistics.Distribution(freqlist)
```

This too offers a dictionary-like interface, where values are by definition normalised. The advantage of a Distribution class is that it offers information-theoretic methods such as entropy(), maxentropy(), perplexity() and poslog().

A frequency list can be saved to file using the <code>save(filename)</code> method, and loaded back from file using the <code>load(filename)</code> method. The <code>output()</code> method is a generator yielding strings for each line of output, in ranked order.

#### 8.3 API Reference

This is a Python library containing classes for Statistic and Information Theoretical computations. It also contains some code from Peter Norvig, AI: A Modern Appproach: http://aima.cs.berkeley.edu/python/utils.html

```
class pynlpl.statistics.Distribution(data, base=2)
```

A distribution can be created over a FrequencyList or a plain dictionary with numeric values. It will be normalized automatically. This implementation uses dictionaries/hashing

```
entropy(base=2)
```

Compute the entropy of the distribution

```
information(type)
```

Computes the information content of the specified type:  $-log_e(p(X))$ 

items()

Returns an *unranked* list of (type, prob) pairs. Use this only if you are not interested in the order.

```
keys()
```

```
maxentropy (base=2)
```

Compute the maximum entropy of the distribution: log\_e(N)

mode (

Returns the type that occurs the most frequently in the probability distribution

```
output (delimiter='\t', freqlist=None)
```

Generator yielding formatted strings expressing the time and probabily for each item in the distribution

```
perplexity (base=2)
```

```
poslog(type)
```

alias for information content

values()

```
class pynlpl.statistics.FrequencyList (tokens=None,
                                                                        casesensitive=True,
                                                                                                dovalida-
                                                     tion=True)
     A frequency list (implemented using dictionaries)
     append (tokens)
           Add a list of tokens to the frequencylist. This method will count them for you.
     count (type, amount=1)
           Count a certain type. The counter will increase by the amount specified (defaults to one)
     dict()
     items()
           Returns an unranked list of (type, count) pairs. Use this only if you are not interested in the order.
     load (filename)
           Load a frequency list from file (in the format produced by the save method)
     mode()
           Returns the type that occurs the most frequently in the frequency list
     output (delimiter='\t', addnormalised=False)
           Print a representation of the frequency list
     p (type)
           Returns the probability (relative frequency) of the token
     save (filename, addnormalised=False)
           Save a frequency list to file, can be loaded later using the load method
     sum()
           Returns the total amount of tokens
     tokens()
           Returns the total amount of tokens
     typetokenratio()
           Computes the type/token ratio
     values()
class pynlpl.statistics.HiddenMarkovModel (startstate, endstate=None)
     print_dptable(V)
     setemission (state, distribution)
     viterbi (observations, doprint=False)
class pynlpl.statistics.MarkovChain(startstate, endstate=None)
     accessible (fromstate, tostate)
           Is state tonode directly accessible (in one step) from state from node? (i.e. is there an edge between the
           nodes). If so, return the probability, else zero
     communicates (fromstate, tostate, maxlength=999999)
           See if a node communicates (directly or indirectly) with another. Returns the probability of the shortest
           path (probably, but not necessarily the highest probability)
     p (sequence, subsequence=True)
           Returns the probability of the given sequence or subsequence (if subsequence=True, default).
     reducible()
```

8.3. API Reference 1063

```
settransitions (state, distribution)
     size()
pynlpl.statistics.dotproduct(X, Y)
     Return the sum of the element-wise product of vectors x and y. >>> dotproduct([1, 2, 3], [1000, 100, 10]) 1230
pynlpl.statistics.histogram (values, mode=0, bin function=None)
     Return a list of (value, count) pairs, summarizing the input values. Sorted by increasing value, or if mode=1, by
     decreasing count. If bin function is given, map it over values first.
pynlpl.statistics.levenshtein(s1, s2, maxdistance=9999)
     Computes the levenshtein distance between two strings. Adapted from: http://en.wikibooks.org/wiki/
     Algorithm_Implementation/Strings/Levenshtein_distance#Python
pynlpl.statistics.log2(x)
     Base 2 logarithm. >>> log2(1024) 10.0
pynlpl.statistics.mean (values)
     Return the arithmetic average of the values.
pynlpl.statistics.median(values)
     Return the middle value, when the values are sorted. If there are an odd number of elements, try to average the
     middle two. If they can't be averaged (e.g. they are strings), choose one at random. >>> median([10, 100, 11])
     11 >>> median([1, 2, 3, 4]) 2.5
pynlpl.statistics.mode (values)
     Return the most common value in the list of values. \gg \mod([1, 2, 3, 2])
pynlpl.statistics.normalize(numbers, total=1.0)
     Multiply each number by a constant such that the sum is 1.0 (or total). >>> normalize([1,2,1]) [0.25, 0.5, 0.25]
pynlpl.statistics.product (seq)
     Return the product of a sequence of numerical values. >>> product([1,2,6]) 12
pynlpl.statistics.stddev(values, meanval=None)
     The standard deviation of a set of values. Pass in the mean if you already know it.
pynlpl.statistics.vector_add(a, b)
     Component-wise addition of two vectors. >>> vector\_add((0, 1), (8, 9)) (8, 10)
```

# CHAPTER 9

**Text Processors** 

This module contains classes and functions for text processing. It is imported as follows:

```
import pynlpl.textprocessors
```

## 9.1 Tokenisation

A very crude tokeniser is available in the form of the function <code>pynlpl.textprocessors.crude\_tokeniser(string)</code>. This will split punctuation characters from words and returns a list of tokens. It however has no regard for abbreviations and end-of-sentence detection, which is functionality a more sophisticated tokeniser can provide:

```
tokens = pynlpl.textprocessors.crude_tokeniser("to be, or not to be.")
```

This will result in:

```
tokens == ['to','be',',','or','not','to','be','.']
```

## 9.2 N-gram extraction

The extraction of n-grams is an elemental operation in Natural Language Processing. PyNLPI offers the Windower class to accomplish this task:

The input to the Windower should be a list of words and a value for n. In addition, the windower can output extra symbols at the beginning of the input sequence and at the end of it. By default, this behaviour is enabled and the input symbol is <br/>begin>, whereas the output symbol is <end>. If this behaviour is unwanted you can suppress it by instantiating the Windower as follows:

```
Windower(tokens, 3, None, None)
```

The Windower is implemented as a Python generator and at each iteration yields a tuple of length n.

```
class pynlpl.textprocessors.MultiWindower(tokens, min_n=1, max_n=9, begin-
marker=None, endmarker=None)
```

Extract n-grams of various configurations from a sequence

```
class pynlpl.textprocessors.ReflowText (stream, filternontext=True)
```

Attempts to re-flow a text that has arbitrary line endings in it. Also undoes hyphenisation

```
class pynlpl.textprocessors.Tokenizer(stream, splitsentences=True,
```

onesentenceperline=False, regexps=(re.compile('^(?:(?:https?):(?:(?://)|(?:\\\))|www\)(?:[\w\d:#@%/;\$ =\\\ &](?:#!)?)\*'), re.compile('^[A-Za-z0-9\\+\_-]+@[A-Za-z0-9\\_\_-]+(?:\[a-zA-Z]+)+')))

A tokenizer and sentence splitter, which acts on a file/stream-like object and when iterating over the object it yields a lists of tokens (in case the sentence splitter is active (default)), or a token (if the sentence splitter is deactivated).

```
class pynlpl.textprocessors.Windower (tokens, n=1, beginmarker='<br/>begin>', end-marker='<end>')
```

Moves a sliding window over a list of tokens, upon iteration in yields all n-grams of specified size in a tuple.

Example without markers:

Example with default markers:

```
>>> for ngram in Windower("This is a test .",3):
...     print(" ".join(ngram))
<begin> <begin> This
<begin> This is
This is a
     is a test
a test .
test . <end>
. <end> <end>
```

```
pynlpl.textprocessors.calculate_overlap(haystack, needle, allowpartial=True)
```

Calculate the overlap between two sequences. Yields (overlap, placement) tuples (multiple because there may be multiple overlaps!). The former is the part of the sequence that overlaps, and the latter is -1 if the overlap is on the left side, 0 if it is a subset, 1 if it overlaps on the right side, 2 if its an identical match

```
pynlpl.textprocessors.crude_tokenizer(text)
    Replaced by tokenize(). Alias
```

```
pynlpl.textprocessors.find_keyword_in_context (tokens, keyword, contextsize=1)
```

Find a keyword in a particular sequence of tokens, and return the local context. Contextsize is the number of words to the left and right. The keyword may have multiple word, in which case it should to passed as a tuple or list

```
pynlpl.textprocessors.is_end_of_sentence(tokens, i)
```

```
pynlpl.textprocessors.split_sentences (tokens)
```

Split sentences (based on tokenised data), returns sentences as a list of lists of tokens, each sentence is a list of tokens

```
pynlpl.textprocessors.strip_accents(s, encoding='utf-8')
```

Strip characters with diacritics and return a flat ascii representation

```
pynlpl.textprocessors.swap(tokens, maxdist=2)
```

Perform a swap operation on a sequence of tokens, exhaustively swapping all tokens up to the maximum specified distance. This is a subset of all permutations.

```
pynlpl.textprocessors.tokenise(text, regexps=(re.compile('^(?:(?:https?):(?:(?://)|(?:\\\\\))|www\\)(?:[\\w\\d:#@%/;$() =\\\\\&](?:#!)?)*'), re.compile('^[A-Za-z0-9\\\+_-]+@[A-Za-z0-9\\\-]+(?:\\[a-zA-Z]+)+')))
```

Alias for the British

```
pynlpl.textprocessors.tokenize(text, regexps=(re.compile('^(?:(?:https?):(?:(?://)|(?:\\\\\))|www\\)(?:[\\w\\d:#@%/;$() =\\\\\&](?:#!)?)*'), re.compile('^[A-Za-z0-9\\\+_-]+@[A-Za-z0-9\\\-]+(?:\\[a-zA-Z]+)+')))
```

Tokenizes a string and returns a list of tokens

#### **Parameters**

- **text** (*string*) The text to tokenise
- regexps (Tuple/list of regular expressions to use in tokenisation) Regular expressions to use as tokeniser rules in tokenisation (default=\_pynlpl.textprocessors.TOKENIZERRULES\_)

Return type Returns a list of tokens

#### Examples:

```
>>> for token in tokenize("This is a test."):
... print(token)
This
is
a
test
...
```

1068

# CHAPTER 10

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## р

```
pynlpl.common, 3
pynlpl.datatypes, 5
pynlpl.evaluation, 9
pynlpl.formats.cgn, 1053
pynlpl.formats.folia,11
pynlpl.formats.giza, 1053
pynlpl.formats.moses, 1054
pynlpl.formats.sonar, 1054
pynlpl.formats.taggerdata, 1055
{\tt pynlpl.formats.timbl, 1055}
pynlpl.lm.client, 1058
pynlpl.lm.lm, 1057
pynlpl.lm.srilm, 1057
pynlpl.search, 1059
pynlpl.statistics, 1062
pynlpl.textprocessors, 1066
```

1072 Python Module Index

#### **Symbols** method), 670 \_init\_\_() (pynlpl.formats.folia.Dependency method), \_\_init\_\_() (pynlpl.formats.folia.AbstractAnnotationLayer method), 75 (pynlpl.formats.folia.DependencyDependent \_init\_\_() (pynlpl.formats.folia.AbstractElement \_\_init\_\_() method), 776 method), 26 (pynlpl.formats.folia.Description method), \_init\_\_() \_\_init\_\_() (pynlpl.formats.folia.AbstractSpanAnnotation 1032 method), 52 \_\_init\_\_() (pynlpl.formats.folia.Division method), 127 init () (pynlpl.formats.folia.AbstractStructureElement \_\_init\_\_() (pynlpl.formats.folia.Document method), 15, method), 38 16 (pynlpl.formats.folia.AbstractTextMarkup \_\_init\_\_() (pynlpl.formats.folia.DomainAnnotation \_init\_\_() method), 87 method), 429 \_\_init\_\_() (pynlpl.formats.folia.AbstractTokenAnnotation init () (pynlpl.formats.folia.EnddatetimeFeature method), 64 method), 907 init () (pynlpl.formats.folia.ActorFeature method), \_init\_\_() (pynlpl.formats.folia.EntitiesLayer method), 682 \_\_init\_\_() (pynlpl.formats.folia.AlignReference method), init () (pynlpl.formats.folia.Entity method), 553 init () (pynlpl.formats.folia.Entry method), 140 \_\_init\_\_() (pynlpl.formats.folia.Alignment method), 1009 \_init\_\_() (pynlpl.formats.folia.ErrorDetection method), \_\_init\_\_() (pynlpl.formats.folia.AllowTokenAnnotation 965 method), 48 \_\_init\_\_() (pynlpl.formats.folia.Event method), 153 \_\_init\_\_() (pynlpl.formats.folia.Alternative method), 919 \_\_init\_\_() (pynlpl.formats.folia.Example method), 166 \_\_init\_\_() (pynlpl.formats.folia.AlternativeLayers \_\_init\_\_() (pynlpl.formats.folia.Feature method), 863 method), 932 \_\_init\_\_() (pynlpl.formats.folia.Figure method), 179 \_init\_\_() (pynlpl.formats.folia.BegindatetimeFeature \_init\_\_() (pynlpl.formats.folia.Gap method), 191 method), 896 \_\_init\_\_() (pynlpl.formats.folia.Head method), 203 init () (pynlpl.formats.folia.Cell method), 101 \_\_init\_\_() (pynlpl.formats.folia.Headspan method), 788 \_\_init\_\_() (pynlpl.formats.folia.Chunk method), 518 (pynlpl.formats.folia.LangAnnotation \_init\_\_() \_\_init\_\_() (pynlpl.formats.folia.ChunkingLayer method), method), 451 647 \_init\_\_() (pynlpl.formats.folia.LemmaAnnotation (pynlpl.formats.folia.CoreferenceChain \_\_init\_\_() method), 462 method), 530 \_init\_\_() (pynlpl.formats.folia.Linebreak method), 216 (pynlpl.formats.folia.CoreferenceLayer init () \_\_init\_\_() (pynlpl.formats.folia.List method), 229 method), 659 \_\_init\_\_() (pynlpl.formats.folia.ListItem method), 242 (pynlpl.formats.folia.CoreferenceLink \_\_init\_\_() init () (pynlpl.formats.folia.Metric method), 1043 method), 765 \_\_init\_\_() (pynlpl.formats.folia.New method), 976 \_\_init\_\_() (pynlpl.formats.folia.Correction method), 944 \_\_init\_\_() (pynlpl.formats.folia.Note method), 255 \_\_init\_\_() (pynlpl.formats.folia.Current method), 954 \_\_init\_\_() (pynlpl.formats.folia.Observation method), \_\_init\_\_() (pynlpl.formats.folia.Definition method), 114 565 \_\_init\_\_() (pynlpl.formats.folia.DependenciesLayer (pynlpl.formats.folia.ObservationLayer init ()

method), 694	753
init() (pynlpl.formats.folia.Original method), 987	init() (pynlpl.formats.folia.Whitespace method),
init() (pynlpl.formats.folia.Paragraph method), 268	399, 400
init() (pynlpl.formats.folia.Part method), 281	init() (pynlpl.formats.folia.Word method), 413
init() (pynlpl.formats.folia.PhonContent method),	init() (pynlpl.formats.fql.Query method), 803
506	iter() (pynlpl.formats.folia.AbstractAnnotationLayer
init() (pynlpl.formats.folia.PosAnnotation method),	method), 83
440	iter() (pynlpl.formats.folia.AbstractElement
init() (pynlpl.formats.folia.Predicate method), 576	method), 34
init() (pynlpl.formats.folia.Quote method), 294	iter() (pynlpl.formats.folia.AbstractSpanAnnotation
init() (pynlpl.formats.folia.Reader method), 805	method), 61
init() (pynlpl.formats.folia.Reference method), 307	iter() (pynlpl.formats.folia.AbstractStructureElement
init() (pynlpl.formats.folia.Row method), 320	method), 47
init() (pynlpl.formats.folia.SemanticRole method),	iter() (pynlpl.formats.folia.AbstractTextMarkup
623	method), 94
init() (pynlpl.formats.folia.SemanticRolesLayer	iter() (pynlpl.formats.folia.AbstractTokenAnnotation
method), 741	method), 72
init() (pynlpl.formats.folia.SenseAnnotation method), 473	iter() (pynlpl.formats.folia.ActorFeature method), 893
init() (pynlpl.formats.folia.Sentence method), 333	iter() (pynlpl.formats.folia.AlignReference method),
init() (pynlpl.formats.folia.Sentiment method), 588	1028
init() (pynlpl.formats.folia.Sentiment incured), 566init() (pynlpl.formats.folia.SentimentLayer method),	iter() (pynlpl.formats.folia.Alignment method), 1017
706	iter() (pynlpl.formats.folia.Alternative method), 928
init() (pynlpl.formats.folia.Statement method), 600	iter() (pynlpl.formats.folia.AlternativeLayers
init() (pynlpl.formats.folia.StatementLayer method),	method), 939
717	iter() (pynlpl.formats.folia.BegindatetimeFeature
init() (pynlpl.formats.folia.SubjectivityAnnotation	method), 904
method), 484	iter() (pynlpl.formats.folia.Cell method), 110
init() (pynlpl.formats.folia.Suggestion method), 998	iter() (pynlpl.formats.folia.Chunk method), 527
init() (pynlpl.formats.folia.SynsetFeature method),	iter() (pynlpl.formats.folia.ChunkingLayer method),
874	655
init() (pynlpl.formats.folia.SyntacticUnit method),	iter() (pynlpl.formats.folia.CoreferenceChain
611	method), 538
init() (pynlpl.formats.folia.SyntaxLayer method),	iter() (pynlpl.formats.folia.CoreferenceLayer
729	method), 667
init() (pynlpl.formats.folia.Table method), 347	iter() (pynlpl.formats.folia.CoreferenceLink
init() (pynlpl.formats.folia.TableHead method), 373	method), 773
init() (pynlpl.formats.folia.Term method), 360 init() (pynlpl.formats.folia.Text method), 386	iter() (pynlpl.formats.folia.Correction method), 951 iter() (pynlpl.formats.folia.Current method), 962
init() (pynlpl.formats.folia.Text method), 380init() (pynlpl.formats.folia.TextContent method),	iter() (pynlpl.formats.folia.Current method), 902 iter() (pynlpl.formats.folia.Definition method), 123
int() (pyinpi.formats.fona.fextContent inlethod),	iter() (pynlpl.formats.folia.DependenciesLayer
init() (pynlpl.formats.folia.TextMarkupCorrection	method), 679
method), 841	iter() (pynlpl.formats.folia.Dependency method),
init() (pynlpl.formats.folia.TextMarkupError	550
method), 851	iter() (pynlpl.formats.folia.DependencyDependent
init() (pynlpl.formats.folia.TextMarkupGap	method), 785
method), 809	iter() (pynlpl.formats.folia.Description method),
init() (pynlpl.formats.folia.TextMarkupString	1040
method), 819	iter() (pynlpl.formats.folia.Division method), 136
init() (pynlpl.formats.folia.TextMarkupStyle	iter() (pynlpl.formats.folia.DomainAnnotation
method), 830	method), 437
init() (pynlpl.formats.folia.TimeSegment method),	iter() (pynlpl.formats.folia.EnddatetimeFeature
635	method), 915
init() (pynlpl.formats.folia.TimingLayer method),	iter() (pynlpl.formats.folia.EntitiesLayer method),

1074 Index

691	iter() (pynlpl.formats.folia.SynsetFeature method),
iter() (pynlpl.formats.folia.Entity method), 561	882
iter() (pynlpl.formats.folia.Entry method), 149	iter() (pynlpl.formats.folia.SyntacticUnit method),
iter() (pynlpl.formats.folia.ErrorDetection method),	620
973	iter() (pynlpl.formats.folia.SyntaxLayer method),
iter() (pynlpl.formats.folia.Event method), 162	738
iter() (pynlpl.formats.folia.Example method), 175	iter() (pynlpl.formats.folia.Table method), 357
iter() (pynlpl.formats.folia.Feature method), 871	iter() (pynlpl.formats.folia.TableHead method), 383
iter() (pynlpl.formats.folia.Figure method), 188	iter() (pynlpl.formats.folia.Term method), 370
iter() (pynlpl.formats.folia.Gap method), 199	iter() (pynlpl.formats.folia.Text method), 396
iter() (pynlpl.formats.folia.Head method), 212	iter() (pynlpl.formats.folia.TextContent method),
iter() (pynlpl.formats.folia.Headspan method), 796	503
iter() (pynlpl.formats.folia.LangAnnotation method), 459	iter() (pynlpl.formats.folia.TextMarkupCorrection method), 848
iter() (pynlpl.formats.folia.LemmaAnnotation	iter() (pynlpl.formats.folia.TextMarkupError
method), 470	method), 859
iter() (pynlpl.formats.folia.Linebreak method), 225	iter() (pynlpl.formats.folia.TextMarkupGap
iter() (pynlpl.formats.folia.List method), 238	method), 816
iter() (pynlpl.formats.folia.ListItem method), 251	iter() (pynlpl.formats.folia.TextMarkupString
iter() (pynlpl.formats.folia.Metric method), 1051	method), 827
iter() (pynlpl.formats.folia.New method), 984	iter() (pynlpl.formats.folia.TextMarkupStyle
iter() (pynlpl.formats.folia.Note method), 264	method), 838
iter() (pynlpl.formats.folia.Observation method), 573	iter() (pynlpl.formats.folia.TimeSegment method), 643
iter() (pynlpl.formats.folia.ObservationLayer	iter() (pynlpl.formats.folia.TimingLayer method),
method), 702	761
iter() (pynlpl.formats.folia.Original method), 995	iter() (pynlpl.formats.folia.Whitespace method), 409
iter() (pynlpl.formats.folia.Paragraph method), 277	iter() (pynlpl.formats.folia.Word method), 424
iter() (pynlpl.formats.folia.Part method), 290	len() (pynlpl.formats.folia.AbstractAnnotationLayer
iter() (pynlpl.formats.folia.PhonContent method),	method), 84
513	len() (pynlpl.formats.folia.AbstractElement
iter() (pynlpl.formats.folia.PosAnnotation method),	method), 34
448	len() (pynlpl.formats.folia.AbstractSpanAnnotation
iter() (pynlpl.formats.folia.Predicate method), 585	method), 61
iter() (pynlpl.formats.folia.Quote method), 303	len() (pynlpl.formats.folia.AbstractStructureElement
iter() (pynlpl.formats.folia.Reference method), 316	method), 47
iter() (pynlpl.formats.folia.Row method), 329	len() (pynlpl.formats.folia.AbstractTextMarkup method), 94
iter() (pynlpl.formats.folia.SemanticRole method), 631	len() (pynlpl.formats.folia.AbstractTokenAnnotation
iter() (pynlpl.formats.folia.SemanticRolesLayer	method), 72
method), 749	len() (pynlpl.formats.folia.ActorFeature method),
iter() (pynlpl.formats.folia.SenseAnnotation	893
method), 481	len() (pynlpl.formats.folia.AlignReference method),
iter() (pynlpl.formats.folia.Sentence method), 344	1029
iter() (pynlpl.formats.folia.Sentiment method), 596	len() (pynlpl.formats.folia.Alignment method), 1017
iter() (pynlpl.formats.folia.SentimentLayer method),	len() (pynlpl.formats.folia.Alternative method), 929
714	len() (pynlpl.formats.folia.AlternativeLayers
iter() (pynlpl.formats.folia.Statement method), 608	method), 940
iter() (pynlpl.formats.folia.StatementLayer method),	len() (pynlpl.formats.folia.BegindatetimeFeature
726	method), 904
iter() (pynlpl.formats.folia.SubjectivityAnnotation	len() (pynlpl.formats.folia.Cell method), 110
method), 492	len() (pynlpl.formats.folia.Chunk method), 527
iter() (pynlpl.formats.folia.Suggestion method),	len() (pynlpl.formats.folia.ChunkingLayer method),
1006	655

Index 1075

len() (pynlpl.formats.folia.CoreferenceChain	448
method), 538	len() (pynlpl.formats.folia.Predicate method), 585
len() (pynlpl.formats.folia.CoreferenceLayer	len() (pynlpl.formats.folia.Quote method), 303
method), 667	len() (pynlpl.formats.folia.Reference method), 316
len() (pynlpl.formats.folia.CoreferenceLink	len() (pynlpl.formats.folia.Row method), 329
method), 773	len() (pynlpl.formats.folia.SemanticRole method).
len() (pynlpl.formats.folia.Correction method), 951	631
len() (pynlpl.formats.folia.Current method), 962	
len() (pynlpl.formats.folia.Definition method), 123	method), 750
len() (pynlpl.formats.folia.DependenciesLayer	len() (pynlpl.formats.folia.SenseAnnotation
method), 679	method), 481
len() (pynlpl.formats.folia.Dependency method),	len() (pynlpl.formats.folia.Sentence method), 344
550	len() (pynlpl.formats.folia.Sentiment method), 596
len() (pynlpl.formats.folia.DependencyDependent method), 785	len() (pynlpl.formats.folia.SentimentLayer method), 714
len() (pynlpl.formats.folia.Description method),	len() (pynlpl.formats.folia.Statement method), 608
1040	len() (pynlpl.formats.folia.StatementLayer method),
len() (pynlpl.formats.folia.Division method), 136	726
len() (pynlpl.formats.folia.DomainAnnotation	len() (pynlpl.formats.folia.SubjectivityAnnotation
method), 437	method), 492
len() (pynlpl.formats.folia.EnddatetimeFeature	len() (pynlpl.formats.folia.Suggestion method),
method), 915	1006
len() (pynlpl.formats.folia.EntitiesLayer method), 691	len() (pynlpl.formats.folia.SynsetFeature method),
len() (pynlpl.formats.folia.Entity method), 562	
	len() (pynlpl.formats.folia.SyntacticUnit method),
len() (pynlpl.formats.folia.Entry method), 149	
len() (pynlpl.formats.folia.ErrorDetection method),	len() (pynlpl.formats.folia.SyntaxLayer method),
973	738
len() (pynlpl.formats.folia.Event method), 162	len() (pynlpl.formats.folia.Table method), 357
len() (pynlpl.formats.folia.Example method), 175	len() (pynlpl.formats.folia.TableHead method), 383
len() (pynlpl.formats.folia.Feature method), 871	len() (pynlpl.formats.folia.Term method), 370
len() (pynlpl.formats.folia.Figure method), 188	len() (pynlpl.formats.folia.Text method), 396
len() (pynlpl.formats.folia.Gap method), 199	len() (pynlpl.formats.folia.TextContent method), 503
len() (pynlpl.formats.folia.Head method), 212	len() (pynlpl.formats.folia.TextMarkupCorrection
len() (pynlpl.formats.folia.Headspan method), 796	method), 848
len() (pynlpl.formats.folia.LangAnnotation method),	len() (pynlpl.formats.folia.TextMarkupError
459	method), 859
len() (pynlpl.formats.folia.LemmaAnnotation	len() (pynlpl.formats.folia.TextMarkupGap method).
method), 470	816
len() (pynlpl.formats.folia.Linebreak method), 225	len() (pynlpl.formats.folia.TextMarkupString
len() (pynlpl.formats.folia.List method), 238	method), 827
len() (pynlpl.formats.folia.ListItem method), 251	len() (pynlpl.formats.folia.TextMarkupStyle
len() (pynlpl.formats.folia.Metric method), 1051	method), 838
len() (pynlpl.formats.folia.New method), 984	len() (pynlpl.formats.folia.TimeSegment method),
len() (pynlpl.formats.folia.Note method), 264	643
len() (pynlpl.formats.folia.Observation method), 573	1 0 ( 110 . 01
len() (pynlpl.formats.folia.ObservationLayer	761
method), 703	len() (pynlpl.formats.folia.Whitespace method), 409
len() (pynlpl.formats.folia.Original method), 995	len() (pynlpl.formats.folia.Word method), 424
len() (pynlpl.formats.folia.Paragraph method), 277	str() (pynlpl.formats.folia.AbstractAnnotationLayer
len() (pynlpl.formats.folia.Part method), 290	method), 84
len() (pynlpl.formats.folia.PhonContent method),	str() (pynlpl.formats.folia.AbstractElement method).
514	34
len() (pynlpl.formats.folia.PosAnnotation method),	str() (pynlpl.formats.folia.AbstractSpanAnnotation

1076 Index

method), 61	str() (pynlpl.formats.folia.Gap method), 199
str() (pynlpl.formats.folia.AbstractStructureElement method), 47	str() (pynlpl.formats.folia.Head method), 212 str() (pynlpl.formats.folia.Headspan method), 797
str() (pynlpl.formats.folia.AbstractTextMarkup	str() (pynlpl.formats.folia.LangAnnotation method),
method), 94	459
str() (pynlpl.formats.folia.AbstractTokenAnnotation	str() (pynlpl.formats.folia.LemmaAnnotation
method), 72	method), 470
str() (pynlpl.formats.folia.ActorFeature method),	str() (pynlpl.formats.folia.Linebreak method), 225
893	str() (pynlpl.formats.folia.List method), 238
str() (pynlpl.formats.folia.AlignReference method),	str() (pynlpl.formats.folia.ListItem method), 251
1029	str() (pynlpl.formats.folia.Metric method), 1051
str() (pynlpl.formats.folia.Alignment method), 1017	str() (pynlpl.formats.folia.New method), 984
str() (pynlpl.formats.folia.AllowTokenAnnotation	str() (pynlpl.formats.folia.Note method), 264
method), 49	str() (pynlpl.formats.folia.Observation method), 573
str() (pynlpl.formats.folia.Alternative method), 929	str() (pynlpl.formats.folia.ObservationLayer
str() (pynlpl.formats.folia.AlternativeLayers	method), 703
method), 940	str() (pynlpl.formats.folia.Original method), 995
str() (pynlpl.formats.folia.BegindatetimeFeature method), 904	str() (pynlpl.formats.folia.Paragraph method), 277 str() (pynlpl.formats.folia.Part method), 290
str() (pynlpl.formats.folia.Cell method), 110	str() (pynlpl.formats.folia.PhonContent method),
str() (pynlpl.formats.folia.Cen method), 527	su() (pyinpi.formats.form.r noncontent inchod),
str() (pynlpl.formats.folia.ChunkingLayer method),	str() (pynlpl.formats.folia.PosAnnotation method),
655	448
str() (pynlpl.formats.folia.CoreferenceChain	str() (pynlpl.formats.folia.Predicate method), 585
method), 538	str() (pynlpl.formats.folia.Quote method), 303
str() (pynlpl.formats.folia.CoreferenceLayer	str() (pynlpl.formats.folia.Reference method), 316
method), 667	str() (pynlpl.formats.folia.Row method), 329
str() (pynlpl.formats.folia.CoreferenceLink method),	str() (pynlpl.formats.folia.SemanticRole method),
str() (pynlpl.formats.folia.Correction method), 951	str() (pynlpl.formats.folia.SemanticRolesLayer
str() (pynlpl.formats.folia.Current method), 962	method), 750
str() (pynlpl.formats.folia.Definition method), 123	str() (pynlpl.formats.folia.SenseAnnotation method),
str() (pynlpl.formats.folia.DependenciesLayer	481
method), 679	str() (pynlpl.formats.folia.Sentence method), 344
str() (pynlpl.formats.folia.Dependency method), 550	str() (pynlpl.formats.folia.Sentiment method), 597
str() (pynlpl.formats.folia.DependencyDependent	str() (pynlpl.formats.folia.SentimentLayer method),
method), 785	714
str() (pynlpl.formats.folia.Description method),	str() (pynlpl.formats.folia.Statement method), 608
1040	str() (pynlpl.formats.folia.StatementLayer method),
str() (pynlpl.formats.folia.Division method), 136	726
str() (pynlpl.formats.folia.DomainAnnotation method), 437	str() (pynlpl.formats.folia.SubjectivityAnnotation method), 492
str() (pynlpl.formats.folia.EnddatetimeFeature	str() (pynlpl.formats.folia.Suggestion method), 1006
method), 915	str() (pynlpl.formats.folia.SynsetFeature method),
str() (pynlpl.formats.folia.EntitiesLayer method),	882
691	str() (pynlpl.formats.folia.SyntacticUnit method),
str() (pynlpl.formats.folia.Entity method), 562	620
str() (pynlpl.formats.folia.Entry method), 149	str() (pynlpl.formats.folia.SyntaxLayer method), 738
str() (pynlpl.formats.folia.ErrorDetection method),	str() (pynlpl.formats.folia.Table method), 357
973	str() (pynlpl.formats.folia.TableHead method), 383
str() (pynlpl.formats.folia.Event method), 162	str() (pynlpl.formats.folia.Term method), 370
str() (pynlpl.formats.folia.Example method), 175	str() (pynlpl.formats.folia.Text method), 396
str() (pynlpl.formats.folia.Feature method), 871 str() (pynlpl.formats.folia.Figure method), 188	str() (pynlpl.formats.folia.TextContent method), 503
su() (pympi.ioimais.ioma.i iguie memou), 100	

str() (pynlpl.formats.folia.TextMarkupCorrection method), 848	ACCEPTED_DATA (pynlpl.formats.folia.BegindatetimeFeature attribute), 895
str() (pynlpl.formats.folia.TextMarkupError method), 859	ACCEPTED_DATA (pynlpl.formats.folia.Cell attribute), 100
str() (pynlpl.formats.folia.TextMarkupGap method), 816	ACCEPTED_DATA (pynlpl.formats.folia.Chunk attribute), 518
str() (pynlpl.formats.folia.TextMarkupString method), 827	ACCEPTED_DATA (pynlpl.formats.folia.ChunkingLayer attribute), 646
str() (pynlpl.formats.folia.TextMarkupStyle method), 838	ACCEPTED_DATA (pynlpl.formats.folia.CoreferenceChain attribute), 529
str() (pynlpl.formats.folia.TimeSegment method),	ACCEPTED_DATA (pynlpl.formats.folia.CoreferenceLayer attribute), 658
str() (pynlpl.formats.folia.TimingLayer method),	ACCEPTED_DATA (pynlpl.formats.folia.CoreferenceLink attribute), 764
str() (pynlpl.formats.folia.Whitespace method), 409 str() (pynlpl.formats.folia.Word method), 424	ACCEPTED_DATA (pynlpl.formats.folia.Correction attribute), 944
A	ACCEPTED_DATA (pynlpl.formats.folia.Current attribute), 953
AbstractAnnotationLayer (class in pynlpl.formats.folia),	ACCEPTED_DATA (pynlpl.formats.folia.Definition attribute), 113
AbstractElement (class in pynlpl.formats.folia), 23 AbstractExperiment (class in pynlpl.evaluation), 9	ACCEPTED_DATA (pynlpl.formats.folia.DependenciesLayer attribute), 670
AbstractSearch (class in pynlpl.search), 1059 AbstractSearchState (class in pynlpl.search), 1059	ACCEPTED_DATA (pynlpl.formats.folia.Dependency attribute), 541
AbstractSpanAnnotation (class in pynlpl.formats.folia),	ACCEPTED_DATA (pynlpl.formats.folia.DependencyDependent attribute), 776
AbstractStructureElement (class in pynlpl.formats.folia), 34	ACCEPTED_DATA (pynlpl.formats.folia.Description attribute), 1031
AbstractTextMarkup (class in pynlpl.formats.folia), 84 AbstractTokenAnnotation (class in pynlpl.formats.folia),	ACCEPTED_DATA (pynlpl.formats.folia.Division attribute), 126
61 ACCEPTED_DATA (pynlpl.formats.folia.AbstractAnnotati	ACCEPTED_DATA (pynlpl.formats.folia.DomainAnnotation
attribute), 74	ACCEPTED_DATA (pynlpl.formats.folia.EnddatetimeFeature
ACCEPTED_DATA (pynlpl.formats.folia.AbstractElement attribute), 26	ACCEPTED_DATA (pynlpl.formats.folia.EntitiesLayer
ACCEPTED_DATA (pynlpl.formats.folia.AbstractSpanAnrattribute), 52	ACCEPTED_DATA (pynlpl.formats.folia.Entity at-
ACCEPTED_DATA (pynlpl.formats.folia.AbstractStructure attribute), 37	ACCEPTED_DATA (pynlpl.formats.folia.Entry at-
ACCEPTED_DATA (pynlpl.formats.folia.AbstractTextMarattribute), 86	ACCEPTED_DATA (pynlpl.formats.folia.ErrorDetection
ACCEPTED_DATA (pynlpl.formats.folia.AbstractTokenArattribute), 63	ACCEPTED_DATA (pynlpl.formats.folia.Event at-
ACCEPTED_DATA (pynlpl.formats.folia.ActorFeature attribute), 884	tribute), 152 ACCEPTED_DATA (pynlpl.formats.folia.Example at-
ACCEPTED_DATA (pynlpl.formats.folia.Alignment attribute), 1009	tribute), 165 ACCEPTED_DATA (pynlpl.formats.folia.Feature at-
ACCEPTED_DATA (pynlpl.formats.folia.AlignReference attribute), 1020	tribute), 862 ACCEPTED_DATA (pynlpl.formats.folia.Figure at-
ACCEPTED_DATA (pynlpl.formats.folia.Alternative attribute), 919	tribute), 178 ACCEPTED_DATA (pynlpl.formats.folia.Gap attribute),
ACCEPTED_DATA (pynlpl.formats.folia.AlternativeLayer	s 191
attribute), 931	ACCEPTED_DATA (pynlpl.formats.folia.Head attribute), 202

- ACCEPTED\_DATA (pynlpl.formats.folia.Headspan at ACCEPTED\_DATA (pynlpl.formats.folia.StatementLayer tribute), 787 attribute), 717
- ACCEPTED\_DATA (pynlpl.formats.folia.LangAnnotation ACCEPTED\_DATA (pynlpl.formats.folia.SubjectivityAnnotation attribute), 450 attribute), 483
- ACCEPTED\_DATA (pynlpl.formats.folia.LemmaAnnotationACCEPTED\_DATA (pynlpl.formats.folia.Suggestion atattribute), 461 tribute), 997
- ACCEPTED\_DATA (pynlpl.formats.folia.Linebreak at- ACCEPTED\_DATA (pynlpl.formats.folia.SynsetFeature tribute), 215 attribute), 873
- ACCEPTED\_DATA (pynlpl.formats.folia.List attribute), ACCEPTED\_DATA (pynlpl.formats.folia.SyntacticUnit attribute), 611
- ACCEPTED\_DATA (pynlpl.formats.folia.ListItem ACCEPTED\_DATA (pynlpl.formats.folia.SyntaxLayer attribute), 241 attribute), 729
- ACCEPTED\_DATA (pynlpl.formats.folia.Metric at ACCEPTED\_DATA (pynlpl.formats.folia.Table attribute), 1042 tribute), 347
- ACCEPTED\_DATA (pynlpl.formats.folia.New attribute), ACCEPTED\_DATA (pynlpl.formats.folia.TableHead attribute), 373
- ACCEPTED\_DATA (pynlpl.formats.folia.Note attribute), ACCEPTED\_DATA (pynlpl.formats.folia.Term attribute), 360
- ACCEPTED\_DATA (pynlpl.formats.folia.Observation attribute), 564

  ACCEPTED\_DATA (pynlpl.formats.folia.Text attribute), 386
- ACCEPTED\_DATA (pynlpl.formats.folia.ObservationLayerACCEPTED\_DATA (pynlpl.formats.folia.TextContent attribute), 693 attribute), 495
- attribute), 693

  ACCEPTED\_DATA (pynlpl.formats.folia.Original ACCEPTED\_DATA (pynlpl.formats.folia.TextMarkupCorrection attribute), 986

  attribute), 495

  ACCEPTED\_DATA (pynlpl.formats.folia.TextMarkupCorrection attribute), 840
- ACCEPTED\_DATA (pynlpl.formats.folia.Paragraph at ACCEPTED\_DATA (pynlpl.formats.folia.TextMarkupError tribute), 267 attribute), 851
- ACCEPTED\_DATA (pynlpl.formats.folia.Part attribute), ACCEPTED\_DATA (pynlpl.formats.folia.TextMarkupGap attribute), 808
- ACCEPTED\_DATA (pynlpl.formats.folia.PhonContent ACCEPTED\_DATA (pynlpl.formats.folia.TextMarkupString attribute), 505 attribute), 819
- ACCEPTED\_DATA (pynlpl.formats.folia.PosAnnotation ACCEPTED\_DATA (pynlpl.formats.folia.TextMarkupStyle attribute), 439 attribute), 829
- ACCEPTED\_DATA (pynlpl.formats.folia.Predicate at ACCEPTED\_DATA (pynlpl.formats.folia.TimeSegment tribute), 576 attribute), 634
- ACCEPTED\_DATA (pynlpl.formats.folia.Quote at ACCEPTED\_DATA (pynlpl.formats.folia.TimingLayer tribute), 293 attribute), 752
- ACCEPTED\_DATA (pynlpl.formats.folia.Reference attribute), 306 ACCEPTED\_DATA (pynlpl.formats.folia.Whitespace attribute), 399
- ACCEPTED\_DATA (pynlpl.formats.folia.Row attribute), ACCEPTED\_DATA (pynlpl.formats.folia.Word attribute), 412
- ACCEPTED\_DATA (pynlpl.formats.folia.SemanticRole accepts() (pynlpl.formats.folia.AbstractAnnotationLayer attribute), 622 class method), 75
- ACCEPTED\_DATA (pynlpl.formats.folia.SemanticRolesLa**yec**epts() (pynlpl.formats.folia.AbstractElement class attribute), 740 method), 26
- ACCEPTED\_DATA (pynlpl.formats.folia.SenseAnnotation accepts() (pynlpl.formats.folia.AbstractSpanAnnotation attribute), 472 class method), 52
- ACCEPTED\_DATA (pynlpl.formats.folia.Sentence at accepts() (pynlpl.formats.folia.AbstractStructureElement tribute), 332 class method), 38
- ACCEPTED\_DATA (pynlpl.formats.folia.Sentiment at accepts() (pynlpl.formats.folia.AbstractTextMarkup class method), 87
- ACCEPTED\_DATA (pynlpl.formats.folia.SentimentLayer accepts() (pynlpl.formats.folia.AbstractTokenAnnotation attribute), 705 class method), 64
- ACCEPTED\_DATA (pynlpl.formats.folia.Statement at- accepts() (pynlpl.formats.folia.ActorFeature class tribute), 599 method), 885

- accepts() (pynlpl.formats.folia.Alignment class method), 1009
- (pynlpl.formats.folia.AlignReference accepts() method), 1020
- accepts() (pynlpl.formats.folia.Alternative class method),
- accepts() (pynlpl.formats.folia.AlternativeLayers class method), 932
- accepts() (pynlpl.formats.folia.BegindatetimeFeature class method), 896
- accepts() (pynlpl.formats.folia.Cell class method), 101
- accepts() (pynlpl.formats.folia.Chunk class method), 518
- (pynlpl.formats.folia.ChunkingLayer accepts() method), 647
- accepts() (pynlpl.formats.folia.CoreferenceChain class method), 530
- accepts() (pynlpl.formats.folia.CoreferenceLayer class method), 659
- (pynlpl.formats.folia.CoreferenceLink class accepts() method), 765
- accepts() (pynlpl.formats.folia.Correction class method), 944
- accepts() (pynlpl.formats.folia.Current class method), 954
- accepts() (pynlpl.formats.folia.Definition class method), 114
- accepts() (pynlpl.formats.folia.DependenciesLayer class method), 670
- (pynlpl.formats.folia.Dependency accepts() class method), 542
- (pynlpl.formats.folia.DependencyDependent accepts() class method), 776
- accepts() (pynlpl.formats.folia.Description class method), 1032
- accepts() (pynlpl.formats.folia.Division class method),
- accepts() (pynlpl.formats.folia.DomainAnnotation class method), 429
- accepts() (pynlpl.formats.folia.EnddatetimeFeature class method), 907
- (pynlpl.formats.folia.EntitiesLayer class accepts() method), 682
- accepts() (pynlpl.formats.folia.Entity class method), 553
- accepts() (pynlpl.formats.folia.Entry class method), 140 accepts() (pynlpl.formats.folia.ErrorDetection
- class method), 965
- accepts() (pynlpl.formats.folia.Event class method), 153
- accepts() (pynlpl.formats.folia.Example class method), 166
- accepts() (pynlpl.formats.folia.Feature class method), 863
- accepts() (pynlpl.formats.folia.Figure class method), 179 accepts() (pynlpl.formats.folia.Gap class method), 191
- accepts() (pynlpl.formats.folia.Head class method), 203
- accepts() (pynlpl.formats.folia.Headspan class method), 788

- accepts() (pynlpl.formats.folia.LangAnnotation class method), 451
- accepts() (pynlpl.formats.folia.LemmaAnnotation class method), 462
- accepts() (pynlpl.formats.folia.Linebreak class method), 216
- accepts() (pynlpl.formats.folia.List class method), 229
- accepts() (pynlpl.formats.folia.ListItem class method),
- accepts() (pynlpl.formats.folia.Metric class method),
- accepts() (pynlpl.formats.folia.New class method), 976
- accepts() (pynlpl.formats.folia.Note class method), 255
- (pynlpl.formats.folia.Observation accepts() method), 565
- accepts() (pynlpl.formats.folia.ObservationLayer class method), 694
- accepts() (pynlpl.formats.folia.Original class method),
- accepts() (pynlpl.formats.folia.Paragraph class method),
- accepts() (pynlpl.formats.folia.Part class method), 281
- (pynlpl.formats.folia.PhonContent accepts() class method), 506
- (pynlpl.formats.folia.PosAnnotation accepts() class method), 440
- accepts() (pynlpl.formats.folia.Predicate class method),
- accepts() (pynlpl.formats.folia.Quote class method), 294
- accepts() (pynlpl.formats.folia.Reference class method),
- accepts() (pynlpl.formats.folia.Row class method), 320
- (pynlpl.formats.folia.SemanticRole accepts() method), 623
- accepts() (pynlpl.formats.folia.SemanticRolesLayer class method), 741
- accepts() (pynlpl.formats.folia.SenseAnnotation class method), 473
- accepts() (pynlpl.formats.folia.Sentence class method), 334
- accepts() (pynlpl.formats.folia.Sentiment class method),
- (pynlpl.formats.folia.SentimentLayer accepts() method), 706
- accepts() (pynlpl.formats.folia.Statement class method), 600
- (pynlpl.formats.folia.StatementLayer accepts() method), 717
- (pynlpl.formats.folia.SubjectivityAnnotation accepts() class method), 484
- accepts() (pynlpl.formats.folia.Suggestion class method),
- accepts() (pynlpl.formats.folia.SynsetFeature class method), 874

(pynlpl.formats.folia.SyntacticUnit add() (pynlpl.formats.folia.Chunk method), 518 accepts() method), 611 add() (pynlpl.formats.folia.ChunkingLayer method), 647 (pynlpl.formats.folia.SyntaxLayer add() (pynlpl.formats.folia.CoreferenceChain method), accepts() class method), 729 accepts() (pynlpl.formats.folia.Table class method), 347 add() (pynlpl.formats.folia.CoreferenceLayer method), accepts() (pynlpl.formats.folia.TableHead class method), add() (pynlpl.formats.folia.CoreferenceLink method). accepts() (pynlpl.formats.folia.Term class method), 360 accepts() (pynlpl.formats.folia.Text class method), 387 add() (pynlpl.formats.folia.Correction method), 944 accepts() (pynlpl.formats.folia.TextContent add() (pynlpl.formats.folia.Current method), 954 method), 496 add() (pynlpl.formats.folia.Definition method), 114 (pynlpl.formats.folia.TextMarkupCorrection add() (pynlpl.formats.folia.DependenciesLayer method), accepts() class method), 841 accepts() (pynlpl.formats.folia.TextMarkupError add() (pynlpl.formats.folia.Dependency method), 542 class method), 851 add() (pynlpl.formats.folia.DependencyDependent accepts() (pynlpl.formats.folia.TextMarkupGap class method), 776 method), 809 add() (pynlpl.formats.folia.Description method), 1032 accepts() (pynlpl.formats.folia.TextMarkupString add() (pynlpl.formats.folia.Division method), 127 add() (pynlpl.formats.folia.Document method), 17 method), 819 (pynlpl.formats.folia.TextMarkupStyle add() (pynlpl.formats.folia.DomainAnnotation method), accepts() class method), 830 (pynlpl.formats.folia.TimeSegment class add() (pynlpl.formats.folia.EnddatetimeFeature method), accepts() method), 635 (pynlpl.formats.folia.TimingLayer class add() (pynlpl.formats.folia.EntitiesLayer method), 682 accepts() add() (pynlpl.formats.folia.Entity method), 553 method), 753 accepts() (pynlpl.formats.folia.Whitespace class method), add() (pynlpl.formats.folia.Entry method), 140 400 add() (pynlpl.formats.folia.ErrorDetection method), 965 accepts() (pynlpl.formats.folia.Word class method), 414 add() (pynlpl.formats.folia.Event method), 153 accessible() (pynlpl.statistics.MarkovChain add() (pynlpl.formats.folia.Example method), 166 add() (pynlpl.formats.folia.Feature method), 863 1063 add() (pynlpl.formats.folia.Figure method), 179 accuracy() (pynlpl.evaluation.ClassEvaluation method), 9 ActorFeature (class in pynlpl.formats.folia), 882 add() (pynlpl.formats.folia.Gap method), 191 add() (pynlpl.datatypes.PatternSet method), 5 add() (pynlpl.formats.folia.Head method), 203 add() (pynlpl.formats.folia.AbstractAnnotationLayer add() (pynlpl.formats.folia.Headspan method), 788 method), 75 add() (pynlpl.formats.folia.LangAnnotation method), 451 add() (pynlpl.formats.folia.AbstractElement method), 26 add() (pynlpl.formats.folia.LemmaAnnotation method), add() (pynlpl.formats.folia.AbstractSpanAnnotation 462 method), 52 add() (pynlpl.formats.folia.Linebreak method), 216 (pynlpl.formats.folia.AbstractStructureElement add() (pynlpl.formats.folia.List method), 229 add() method), 38 add() (pynlpl.formats.folia.ListItem method), 242 add() (pynlpl.formats.folia.AbstractTextMarkup method), add() (pynlpl.formats.folia.Metric method), 1043 add() (pynlpl.formats.folia.New method), 976 add() (pynlpl.formats.folia.AbstractTokenAnnotation add() (pynlpl.formats.folia.Note method), 255 method), 64 add() (pynlpl.formats.folia.Observation method), 565 add() (pynlpl.formats.folia.ActorFeature method), 885 add() (pynlpl.formats.folia.ObservationLayer method), add() (pynlpl.formats.folia.Alignment method), 1009 add() (pynlpl.formats.folia.Original method), 987 (pynlpl.formats.folia.AlignReference method), add() 1020 add() (pynlpl.formats.folia.Paragraph method), 268 add() (pynlpl.formats.folia.Alternative method), 919 add() (pynlpl.formats.folia.Part method), 281 add() (pynlpl.formats.folia.AlternativeLayers method), add() (pynlpl.formats.folia.PhonContent method), 506 add() (pynlpl.formats.folia.PosAnnotation method), 440 (pynlpl.formats.folia.BegindatetimeFeature add() (pynlpl.formats.folia.Predicate method), 576 add() add() (pynlpl.formats.folia.Quote method), 294 method), 896 add() (pynlpl.formats.folia.Cell method), 101 add() (pynlpl.formats.folia.Reference method), 307

add() (pynlpl.formats.folia.Row method), 320 add() (pynlpl.formats.folia.SemanticRole method), 623 add() (pynlpl.formats.folia.SemanticRolesLayer method), (pynlpl.formats.folia.SenseAnnotation method), add() add() (pynlpl.formats.folia.Sentence method), 334 add() (pynlpl.formats.folia.Sentiment method), 588 add() (pynlpl.formats.folia.SentimentLayer method), 706 add() (pynlpl.formats.folia.Statement method), 600 add() (pynlpl.formats.folia.StatementLayer method), 717 (pynlpl.formats.folia.SubjectivityAnnotation add() method), 484 add() (pynlpl.formats.folia.Suggestion method), 998 add() (pynlpl.formats.folia.SynsetFeature method), 874 add() (pynlpl.formats.folia.SyntacticUnit method), 611 add() (pynlpl.formats.folia.SyntaxLayer method), 729 add() (pynlpl.formats.folia.Table method), 347 add() (pynlpl.formats.folia.TableHead method), 373 add() (pynlpl.formats.folia.Term method), 360 add() (pynlpl.formats.folia.Text method), 387 add() (pynlpl.formats.folia.TextContent method), 496 (pynlpl.formats.folia.TextMarkupCorrection add() method), 841 (pynlpl.formats.folia.TextMarkupError method), add() add() (pynlpl.formats.folia.TextMarkupGap method), 809 add() (pynlpl.formats.folia.TextMarkupString method), add() (pynlpl.formats.folia.TextMarkupStyle method), add() (pynlpl.formats.folia.TimeSegment method), 635 add() (pynlpl.formats.folia.TimingLayer method), 753 add() (pynlpl.formats.folia.Whitespace method), 400 add() (pynlpl.formats.folia.Word method), 414 addable() (pynlpl.formats.folia.AbstractAnnotationLayer class method), 75 addable() (pynlpl.formats.folia.AbstractElement class method), 26 addable() (pynlpl.formats.folia.AbstractSpanAnnotation class method), 52 addable() (pynlpl.formats.folia.AbstractStructureElement class method), 38 addable() (pynlpl.formats.folia.AbstractTextMarkup class method), 87 addable() (pynlpl.formats.folia.AbstractTokenAnnotation class method), 64 addable() (pynlpl.formats.folia.ActorFeature class method), 885 addable() (pynlpl.formats.folia.Alignment class method), (pynlpl.formats.folia.AlignReference addable() class method), 1020 addable() (pynlpl.formats.folia.Alternative class method), addable() (pynlpl.formats.folia.LemmaAnnotation class

919 addable() (pynlpl.formats.folia.AlternativeLayers class method), 932 addable() (pynlpl.formats.folia.BegindatetimeFeature class method), 896 addable() (pynlpl.formats.folia.Cell class method), 101 addable() (pynlpl.formats.folia.Chunk class method), 518 addable() (pynlpl.formats.folia.ChunkingLayer class method), 647 addable() (pynlpl.formats.folia.CoreferenceChain class method), 530 addable() (pynlpl.formats.folia.CoreferenceLayer class method), 659 addable() (pynlpl.formats.folia.CoreferenceLink class method), 765 addable() (pynlpl.formats.folia.Correction class method), 944 addable() (pynlpl.formats.folia.Current class method), addable() (pynlpl.formats.folia.Definition class method), addable() (pynlpl.formats.folia.DependenciesLayer class method), 670 addable() (pynlpl.formats.folia.Dependency method), 542 addable() (pynlpl.formats.folia.DependencyDependent class method), 777 addable() (pynlpl.formats.folia.Description class method), 1032 addable() (pynlpl.formats.folia.Division class method), 127 addable() (pynlpl.formats.folia.DomainAnnotation class method), 429 addable() (pynlpl.formats.folia.EnddatetimeFeature class method), 907 addable() (pynlpl.formats.folia.EntitiesLayer class method), 682 addable() (pynlpl.formats.folia.Entity class method), 553 addable() (pynlpl.formats.folia.Entry class method), 140 addable() (pynlpl.formats.folia.ErrorDetection method), 965 addable() (pynlpl.formats.folia.Event class method), 153 addable() (pynlpl.formats.folia.Example class method), 166 addable() (pynlpl.formats.folia.Feature class method), addable() (pynlpl.formats.folia.Figure class method), 179 addable() (pynlpl.formats.folia.Gap class method), 191 addable() (pynlpl.formats.folia.Head class method), 203 addable() (pynlpl.formats.folia.Headspan class method), addable() (pynlpl.formats.folia.LangAnnotation class

1082 Index

method), 451

method), 462 addable() (pynlpl.formats.folia.Linebreak class method), addable() (pynlpl.formats.folia.List class method), 229 addable() (pynlpl.formats.folia.ListItem class method), addable() (pynlpl.formats.folia.Metric class method), 1043 addable() (pynlpl.formats.folia.New class method), 976 addable() (pynlpl.formats.folia.Note class method), 255 (pynlpl.formats.folia.Observation class method), 565 addable() (pynlpl.formats.folia.ObservationLayer class method), 694 addable() (pynlpl.formats.folia.Original class method), addable() (pynlpl.formats.folia.Paragraph class method), addable() (pynlpl.formats.folia.Part class method), 281 (pynlpl.formats.folia.PhonContent method), 506 addable() (pynlpl.formats.folia.PosAnnotation class method), 440 addable() (pynlpl.formats.folia.Predicate class method), 577 addable() (pynlpl.formats.folia.Quote class method), 294 addable() (pynlpl.formats.folia.Reference class method), addable() (pynlpl.formats.folia.Row class method), 320 (pynlpl.formats.folia.SemanticRole addable() class method), 623 addable() (pynlpl.formats.folia.SemanticRolesLayer class method), 741 addable() (pynlpl.formats.folia.SenseAnnotation class method), 473 addable() (pynlpl.formats.folia.Sentence class method), addable() (pynlpl.formats.folia.Sentiment class method), 588 (pynlpl.formats.folia.SentimentLayer addable() method), 706 addable() (pynlpl.formats.folia.Statement class method), (pynlpl.formats.folia.StatementLayer addable() class method), 717 (pynlpl.formats.folia.SubjectivityAnnotation addable() class method), 484 addable() (pynlpl.formats.folia.Suggestion class method), (pynlpl.formats.folia.SynsetFeature addable() class method), 874 (pynlpl.formats.folia.SyntacticUnit addable() class method), 612 (pynlpl.formats.folia.SyntaxLayer addable() class

method), 729 addable() (pynlpl.formats.folia.Table class method), 348 addable() (pynlpl.formats.folia.TableHead class method), 374 addable() (pynlpl.formats.folia.Term class method), 361 addable() (pynlpl.formats.folia.Text class method), 387 (pynlpl.formats.folia.TextContent addable() method), 496 addable() (pynlpl.formats.folia.TextMarkupCorrection class method), 841 addable() (pynlpl.formats.folia.TextMarkupError class method), 852 (pynlpl.formats.folia.TextMarkupGap addable() class method), 809 addable() (pynlpl.formats.folia.TextMarkupString class method), 819 addable() (pynlpl.formats.folia.TextMarkupStyle class method), 830 addable() (pynlpl.formats.folia.TimeSegment class method), 635 addable() (pynlpl.formats.folia.TimingLayer class method), 753 addable() (pynlpl.formats.folia.Whitespace class method), 400 addable() (pynlpl.formats.folia.Word class method), 414 addidsuffix() (pynlpl.formats.folia.AbstractAnnotationLayer method), 75 addidsuffix() (pynlpl.formats.folia.AbstractElement method), 27  $addidsuffix () \ (pynlpl. formats. folia. Abstract Span Annotation$ method), 53 addidsuffix() (pynlpl.formats.folia.AbstractStructureElement method), 38 addidsuffix() (pynlpl.formats.folia.AbstractTextMarkup method), 87 addidsuffix() (pynlpl.formats.folia.AbstractTokenAnnotation method), 64 addidsuffix() (pynlpl.formats.folia.ActorFeature method), 886 addidsuffix() (pynlpl.formats.folia.Alignment method), 1010 addidsuffix() (pynlpl.formats.folia.AlignReference method), 1021 addidsuffix() (pynlpl.formats.folia.Alternative method), 920 addidsuffix() (pynlpl.formats.folia.AlternativeLayers method), 932 addidsuffix() (pynlpl.formats.folia.BegindatetimeFeature method), 897 addidsuffix() (pynlpl.formats.folia.Cell method), 101 addidsuffix() (pynlpl.formats.folia.Chunk method), 519 (pynlpl.formats.folia.ChunkingLayer addidsuffix()

Index 1083

addidsuffix()

method), 647

(pynlpl.formats.folia.CoreferenceChain

method), 530	268
$addidsuffix () \\ \qquad (pynlpl.formats.folia. Coreference Layer$	addidsuffix() (pynlpl.formats.folia.Part method), 281
method), 659	addidsuffix() (pynlpl.formats.folia.PhonContent method),
addidsuffix() (pynlpl.formats.folia.CoreferenceLink	506
method), 765	addidsuffix() (pynlpl.formats.folia.PosAnnotation
addidsuffix() (pynlpl.formats.folia.Correction method),	method), 440
945	addidsuffix() (pynlpl.formats.folia.Predicate method),
addidsuffix() (pynlpl.formats.folia.Current method), 954	577
addidsuffix() (pynlpl.formats.folia.Definition method),	addidsuffix() (pynlpl.formats.folia.Quote method), 294
114	addidsuffix() (pynlpl.formats.folia.Reference method),
addidsuffix() (pynlpl.formats.folia.DependenciesLayer	307
method), 671	addidsuffix() (pynlpl.formats.folia.Row method), 320
addidsuffix() (pynlpl.formats.folia.Dependency method),	addidsuffix() (pynlpl.formats.folia.SemanticRole
542	method), 623
addidsuffix() (pynlpl.formats.folia.DependencyDependent	addidsuffix() (pynlpl.formats.folia.SemanticRolesLayer
method), 777	method), 741
addidsuffix() (pynlpl.formats.folia.Description method),	addidsuffix() (pynlpl.formats.folia.SenseAnnotation
1032	method), 473
addidsuffix() (pynlpl.formats.folia.Division method), 127 addidsuffix() (pynlpl.formats.folia.DomainAnnotation	addidsuffix() (pynlpl.formats.folia.Sentence method), 334 addidsuffix() (pynlpl.formats.folia.Sentiment method),
addidsuffix() (pynlpl.formats.folia.DomainAnnotation method), 429	588
addidsuffix() (pynlpl.formats.folia.EnddatetimeFeature	addidsuffix() (pynlpl.formats.folia.SentimentLayer
method), 908	method), 706
addidsuffix() (pynlpl.formats.folia.EntitiesLayer	addidsuffix() (pynlpl.formats.folia.Statement method),
method), 683	600
addidsuffix() (pynlpl.formats.folia.Entity method), 554	addidsuffix() (pynlpl.formats.folia.StatementLayer
addidsuffix() (pynlpl.formats.folia.Entry method), 140	method), 718
addidsuffix() (pynlpl.formats.folia.ErrorDetection	addidsuffix() (pynlpl.formats.folia.SubjectivityAnnotation
method), 965	method), 484
addidsuffix() (pynlpl.formats.folia.Event method), 153	addidsuffix() (pynlpl.formats.folia.Suggestion method),
addidsuffix() (pynlpl.formats.folia.Example method), 166	998
addidsuffix() (pynlpl.formats.folia.Feature method), 863	addidsuffix() (pynlpl.formats.folia.SynsetFeature
addidsuffix() (pynlpl.formats.folia.Figure method), 179	method), 874
addidsuffix() (pynlpl.formats.folia.Gap method), 192	addidsuffix() (pynlpl.formats.folia.SyntacticUnit
addidsuffix() (pynlpl.formats.folia.Head method), 203	method), 612
addidsuffix() (pynlpl.formats.folia.Headspan method),	addidsuffix() (pynlpl.formats.folia.SyntaxLayer method),
788	730
$addidsuffix () \\ \qquad (pynlpl.formats.folia. Lang Annotation$	addidsuffix() (pynlpl.formats.folia.Table method), 348
method), 451	addidsuffix() (pynlpl.formats.folia.TableHead method),
addidsuffix() (pynlpl.formats.folia.LemmaAnnotation	374
method), 462	addidsuffix() (pynlpl.formats.folia.Term method), 361
addidsuffix() (pynlpl.formats.folia.Linebreak method),	addidsuffix() (pynlpl.formats.folia.Text method), 387
216	addidsuffix() (pynlpl.formats.folia.TextContent method),
addidsuffix() (pynlpl.formats.folia.List method), 229	496
addidsuffix() (pynlpl.formats.folia.ListItem method), 242	addidsuffix() (pynlpl.formats.folia.TextMarkupCorrection
addidsuffix() (pynlpl.formats.folia.Metric method), 1043	method), 841
addidsuffix() (pynlpl.formats.folia.New method), 976	addidsuffix() (pynlpl.formats.folia.TextMarkupError
addidsuffix() (pynlpl.formats.folia.Note method), 255	method), 852
addidsuffix() (pynlpl.formats.folia.Observation method),	addidsuffix() (pynlpl.formats.folia.TextMarkupGap
565 addidguffix() (pyplp) formats folio Observation Lavar	method), 809
addidsuffix() (pynlpl.formats.folia.ObservationLayer method), 694	addidsuffix() (pynlpl.formats.folia.TextMarkupString method), 820
addidsuffix() (pynlpl.formats.folia.Original method), 987	addidsuffix() (pynlpl.formats.folia.TextMarkupStyle
addieserma() (pympi.iormaes.ioma.originar memod), 307	addidation, (pyingi.ioinata.iona.icativialkupatyic
addidsuffix() (pynlpl.formats.folia.Paragraph method),	method), 830

- addidsuffix() (pynlpl.formats.folia.TimeSegment method), 635 addidsuffix() (pynlpl.formats.folia.TimingLayer method), addidsuffix() (pynlpl.formats.folia.Whitespace method), 400 addidsuffix() (pynlpl.formats.folia.Word method), 414 addtoindex() (pynlpl.formats.folia.AbstractAnnotationLayeraddtoindex() (pynlpl.formats.folia.Entity method), 554 method), 75 addtoindex() (pynlpl.formats.folia.AbstractElement addtoindex() (pynlpl.formats.folia.AbstractSpanAnnotation addtoindex() (pynlpl.formats.folia.Event method), 153 method), 53 addtoindex() (pynlpl.formats.folia.AbstractStructureElementaddtoindex() (pynlpl.formats.folia.Feature method), 863 method), 38 addtoindex() (pynlpl.formats.folia.AbstractTextMarkup
- method), 87 addtoindex() (pynlpl.formats.folia.AbstractTokenAnnotatiomddtoindex() (pynlpl.formats.folia.Headspan method), method), 64 addtoindex() (pynlpl.formats.folia.ActorFeature method),
- addtoindex() (pynlpl.formats.folia.Alignment method), 1010 addtoindex() (pynlpl.formats.folia.AlignReference method), 1021
- addtoindex() (pynlpl.formats.folia.Alternative method),
- addtoindex() (pynlpl.formats.folia.AlternativeLayers method), 932
- addtoindex() (pynlpl.formats.folia.BegindatetimeFeature method), 897
- addtoindex() (pynlpl.formats.folia.Cell method), 101 addtoindex() (pynlpl.formats.folia.Chunk method), 519 addtoindex() (pynlpl.formats.folia.ChunkingLayer
- (pynlpl.formats.folia.CoreferenceChain addtoindex() method), 530

method), 647

- addtoindex() (pynlpl.formats.folia.CoreferenceLayer method), 659
- (pynlpl.formats.folia.CoreferenceLink addtoindex()
- addtoindex() (pynlpl.formats.folia.Correction method),
- addtoindex() (pynlpl.formats.folia.Current method), 954 addtoindex() (pynlpl.formats.folia.Definition method), 114
- (pynlpl.formats.folia.DependenciesLayer addtoindex() method), 671
- addtoindex() (pynlpl.formats.folia.Dependency method),
- addtoindex() (pynlpl.formats.folia.DependencyDependent method), 777
- addtoindex() (pynlpl.formats.folia.Description method), 1032

- addtoindex() (pynlpl.formats.folia.Division method), 127 addtoindex() (pynlpl.formats.folia.DomainAnnotation method), 429
- (pynlpl.formats.folia.EnddatetimeFeature addtoindex() method), 908
- addtoindex() (pynlpl.formats.folia.EntitiesLayer method),
- addtoindex() (pynlpl.formats.folia.Entry method), 140
- addtoindex() (pynlpl.formats.folia.ErrorDetection method), 965
- addtoindex() (pynlpl.formats.folia.Example method), 166
- addtoindex() (pynlpl.formats.folia.Figure method), 179
- addtoindex() (pynlpl.formats.folia.Gap method), 192 addtoindex() (pynlpl.formats.folia.Head method), 203
- 789
  - (pynlpl.formats.folia.LangAnnotation addtoindex() method), 451
  - addtoindex() (pynlpl.formats.folia.LemmaAnnotation method), 462
  - addtoindex() (pynlpl.formats.folia.Linebreak method), 216
  - addtoindex() (pynlpl.formats.folia.List method), 229
  - addtoindex() (pynlpl.formats.folia.ListItem method), 242 addtoindex() (pynlpl.formats.folia.Metric method), 1043
  - addtoindex() (pynlpl.formats.folia.New method), 976
- addtoindex() (pynlpl.formats.folia.Note method), 255
- addtoindex() (pynlpl.formats.folia.Observation method), 565
- addtoindex() (pynlpl.formats.folia.ObservationLayer method), 694
- addtoindex() (pynlpl.formats.folia.Original method), 987 addtoindex() (pynlpl.formats.folia.Paragraph method), 268
- addtoindex() (pynlpl.formats.folia.Part method), 281 addtoindex() (pynlpl.formats.folia.PhonContent method), 507
- addtoindex() (pynlpl.formats.folia.PosAnnotation method), 440
- (pynlpl.formats.folia.Predicate method), addtoindex()
- addtoindex() (pynlpl.formats.folia.Quote method), 294 addtoindex() (pynlpl.formats.folia.Reference method),
- addtoindex() (pynlpl.formats.folia.Row method), 320 (pynlpl.formats.folia.SemanticRole addtoindex() method), 624
- addtoindex() (pynlpl.formats.folia.SemanticRolesLayer method), 741
- addtoindex() (pynlpl.formats.folia.SenseAnnotation method), 473

addtoindex() (pynlpl.formats.folia.Sentence method), 334 method), 38 addtoindex() (pynlpl.formats.folia.Sentiment method), alternatives() (pynlpl.formats.folia.AllowTokenAnnotation 589 method), 48 addtoindex() (pynlpl.formats.folia.SentimentLayer alternatives() (pynlpl.formats.folia.Alternative method), method), 706 addtoindex() (pynlpl.formats.folia.Statement method), alternatives() (pynlpl.formats.folia.Cell method), 101 600 alternatives() (pvnlpl.formats.folia.ChunkingLaver addtoindex() (pynlpl.formats.folia.StatementLayer method), 647 alternatives() method), 718 (pynlpl.formats.folia.CoreferenceLayer addtoindex() (pynlpl.formats.folia.SubjectivityAnnotation method), 659 method), 484 alternatives() (pynlpl.formats.folia.Definition method), addtoindex() (pynlpl.formats.folia.Suggestion method), 114 alternatives() (pynlpl.formats.folia.DependenciesLayer addtoindex() (pynlpl.formats.folia.SynsetFeature method), 671 method), 874 alternatives() (pynlpl.formats.folia.Division method), 127 (pynlpl.formats.folia.EntitiesLayer addtoindex() (pynlpl.formats.folia.SyntacticUnit alternatives() method), 612 method), 683 addtoindex() (pynlpl.formats.folia.SyntaxLayer method), alternatives() (pynlpl.formats.folia.Entry method), 140 alternatives() (pynlpl.formats.folia.Event method), 153 alternatives() (pynlpl.formats.folia.Example method), 166 addtoindex() (pynlpl.formats.folia.Table method), 348 alternatives() (pynlpl.formats.folia.Figure method), 179 addtoindex() (pynlpl.formats.folia.TableHead method), 374 alternatives() (pynlpl.formats.folia.Head method), 203 addtoindex() (pynlpl.formats.folia.Term method), 361 alternatives() (pynlpl.formats.folia.Linebreak method), addtoindex() (pynlpl.formats.folia.Text method), 387 addtoindex() (pynlpl.formats.folia.TextContent method), alternatives() (pynlpl.formats.folia.List method), 229 alternatives() (pynlpl.formats.folia.ListItem method), 242 addtoindex() (pynlpl.formats.folia.TextMarkupCorrection alternatives() (pynlpl.formats.folia.Note method), 255 method), 841 alternatives() (pynlpl.formats.folia.ObservationLayer (pynlpl.formats.folia.TextMarkupError method), 694 addtoindex() alternatives() (pynlpl.formats.folia.Paragraph method), method), 852 (pynlpl.formats.folia.TextMarkupGap addtoindex() 268 method), 809 alternatives() (pynlpl.formats.folia.Part method), 281 (pynlpl.formats.folia.TextMarkupString alternatives() (pynlpl.formats.folia.Quote method), 294 addtoindex() alternatives() (pynlpl.formats.folia.Reference method), method), 820 (pynlpl.formats.folia.TextMarkupStyle addtoindex() alternatives() (pynlpl.formats.folia.Row method), 320 method), 830 addtoindex() (pynlpl.formats.folia.TimeSegment alternatives() (pynlpl.formats.folia.SemanticRolesLayer method), 635 method), 741 addtoindex() (pynlpl.formats.folia.TimingLayer method), (pynlpl.formats.folia.Sentence method), alternatives() 334 addtoindex() (pynlpl.formats.folia.Whitespace method), alternatives() (pynlpl.formats.folia.SentimentLayer method), 706 addtoindex() (pynlpl.formats.folia.Word method), 414 alternatives() (pynlpl.formats.folia.StatementLayer alias() (pynlpl.formats.folia.Document method), 17 method), 718 align() (pynlpl.formats.taggerdata.Taggerdata method), alternatives() (pynlpl.formats.folia.SyntaxLayer method), 1055 730 Alignment (class in pynlpl.formats.folia), 1006 alternatives() (pynlpl.formats.folia.Table method), 348 alternatives() (pynlpl.formats.folia.TableHead method), AlignReference (class in pynlpl.formats.folia), 1018 AllowTokenAnnotation (class in pynlpl.formats.folia), 47 Alternative (class in pynlpl.formats.folia), 916 alternatives() (pynlpl.formats.folia.Term method), 361 AlternativeLayers (class in pynlpl.formats.folia), 929 alternatives() (pynlpl.formats.folia.Text method), 387 alternatives() (pynlpl.formats.folia.AbstractAnnotationLayealternatives() (pynlpl.formats.folia.TimingLayer method), method), 75 alternatives() (pynlpl.formats.folia.AbstractStructureElemenatternatives() (pynlpl.formats.folia.Whitespace method),

400 alternatives() (pynlpl.formats.folia.Word method), 414 ancestor() (pynlpl.formats.folia.AbstractAnnotationLayer method), 76 (pynlpl.formats.folia.AbstractElement ancestor() method), 27 ancestor() (pynlpl.formats.folia.AbstractSpanAnnotation method), 53 ancestor() (pynlpl.formats.folia.AbstractStructureElement method), 38 ancestor() (pynlpl.formats.folia.AbstractTextMarkup method), 87 ancestor() (pynlpl.formats.folia.AbstractTokenAnnotation method), 64 ancestor() (pynlpl.formats.folia.ActorFeature method), 886 ancestor() (pynlpl.formats.folia.Alignment method), 1010 ancestor() (pynlpl.formats.folia.AlignReference method), ancestor() (pynlpl.formats.folia.Alternative method), 920 (pynlpl.formats.folia.AlternativeLayers ancestor() method), 932 (pynlpl.formats.folia.BegindatetimeFeature ancestor() method), 897 ancestor() (pynlpl.formats.folia.Cell method), 101 ancestor() (pynlpl.formats.folia.Chunk method), 519 ancestor() (pynlpl.formats.folia.ChunkingLayer method), 647 (pynlpl.formats.folia.CoreferenceChain ancestor() method), 530 ancestor() (pynlpl.formats.folia.CoreferenceLayer method), 659 (pynlpl.formats.folia.CoreferenceLink ancestor() method), 765 ancestor() (pynlpl.formats.folia.Correction method), 945 ancestor() (pynlpl.formats.folia.Current method), 954 ancestor() (pynlpl.formats.folia.Definition method), 114 ancestor() (pynlpl.formats.folia.DependenciesLayer method), 671 ancestor() (pynlpl.formats.folia.Dependency method), 542 (pynlpl.formats.folia.DependencyDependent ancestor() method), 777 (pynlpl.formats.folia.Description method), ancestor() ancestor() (pynlpl.formats.folia.Division method), 127 (pynlpl.formats.folia.DomainAnnotation ancestor() method), 429 (pynlpl.formats.folia.EnddatetimeFeature ancestor() method), 908 ancestor() (pynlpl.formats.folia.EntitiesLayer method), ancestor() (pynlpl.formats.folia.Entity method), 554 ancestor() (pynlpl.formats.folia.Entry method), 140

ancestor() (pynlpl.formats.folia.ErrorDetection method), ancestor() (pynlpl.formats.folia.Event method), 153 ancestor() (pynlpl.formats.folia.Example method), 166 ancestor() (pynlpl.formats.folia.Feature method), 863 ancestor() (pynlpl.formats.folia.Figure method), 179 ancestor() (pynlpl.formats.folia.Gap method), 192 ancestor() (pynlpl.formats.folia.Head method), 203 ancestor() (pynlpl.formats.folia.Headspan method), 789 ancestor() (pynlpl.formats.folia.LangAnnotation method), 451 (pynlpl.formats.folia.LemmaAnnotation ancestor() method), 462 ancestor() (pynlpl.formats.folia.Linebreak method), 216 ancestor() (pynlpl.formats.folia.List method), 229 ancestor() (pynlpl.formats.folia.ListItem method), 242 ancestor() (pynlpl.formats.folia.Metric method), 1043 ancestor() (pynlpl.formats.folia.New method), 976 ancestor() (pynlpl.formats.folia.Note method), 255 ancestor() (pynlpl.formats.folia.Observation method), 565 ancestor() (pynlpl.formats.folia.ObservationLayer method), 695 ancestor() (pynlpl.formats.folia.Original method), 987 ancestor() (pynlpl.formats.folia.Paragraph method), 268 ancestor() (pynlpl.formats.folia.Part method), 281 ancestor() (pynlpl.formats.folia.PhonContent method), ancestor() (pynlpl.formats.folia.PosAnnotation method), ancestor() (pynlpl.formats.folia.Predicate method), 577 ancestor() (pynlpl.formats.folia.Quote method), 294 ancestor() (pynlpl.formats.folia.Reference method), 307 ancestor() (pynlpl.formats.folia.Row method), 320 ancestor() (pynlpl.formats.folia.SemanticRole method), ancestor() (pynlpl.formats.folia.SemanticRolesLayer method), 742 ancestor() (pynlpl.formats.folia.SenseAnnotation method), 473 ancestor() (pynlpl.formats.folia.Sentence method), 334 ancestor() (pynlpl.formats.folia.Sentiment method), 589 ancestor() (pynlpl.formats.folia.SentimentLayer method), ancestor() (pynlpl.formats.folia.Statement method), 600 ancestor() (pynlpl.formats.folia.StatementLayer method), ancestor() (pynlpl.formats.folia.SubjectivityAnnotation method), 484 ancestor() (pynlpl.formats.folia.Suggestion method), 998 ancestor() (pynlpl.formats.folia.SynsetFeature method), 875 ancestor() (pynlpl.formats.folia.SyntacticUnit method),

Index 1087

612

ancestor() (pynlpl.formats.folia.SyntaxLayer method), ancestor() (pynlpl.formats.folia.Table method), 348 ancestor() (pynlpl.formats.folia.TableHead method), 374 ancestor() (pynlpl.formats.folia.Term method), 361 ancestor() (pynlpl.formats.folia.Text method), 387 ancestor() (pynlpl.formats.folia.TextContent method), (pynlpl.formats.folia.TextMarkupCorrection ancestor() method), 841 ancestor() (pynlpl.formats.folia.TextMarkupError method), 852 (pynlpl.formats.folia.TextMarkupGap ancestor() method), 809 (pynlpl.formats.folia.TextMarkupString ancestor() method), 820 (pynlpl.formats.folia.TextMarkupStyle ancestor() ancestor() (pynlpl.formats.folia.TimeSegment method), 635 ancestor() (pynlpl.formats.folia.TimingLayer method), ancestor() (pynlpl.formats.folia.Whitespace method), 400 ancestor() (pynlpl.formats.folia.Word method), 414 method), 76 ancestors() (pynlpl.formats.folia.AbstractElement method), 27 ancestors() (pynlpl.formats.folia.AbstractSpanAnnotation method), 53 ancestors() (pynlpl.formats.folia.AbstractStructureElement ancestors() (pynlpl.formats.folia.Gap method), 192 method), 38 (pynlpl.formats.folia.AbstractTextMarkup ancestors() method), 87 ancestors() (pynlpl.formats.folia.AbstractTokenAnnotation method), 64 ancestors() (pynlpl.formats.folia.ActorFeature method), 886 ancestors() (pynlpl.formats.folia.Alignment method), 1010 (pynlpl.formats.folia.AlignReference ancestors() method), 1021 (pynlpl.formats.folia.Alternative method), ancestors() 920 (pynlpl.formats.folia.AlternativeLayers ancestors() method), 932 (pynlpl.formats.folia.BegindatetimeFeature ancestors() method), 897 ancestors() (pynlpl.formats.folia.Cell method), 102 ancestors() (pynlpl.formats.folia.Chunk method), 519 (pynlpl.formats.folia.ChunkingLayer ancestors() method), 648 (pynlpl.formats.folia.CoreferenceChain ancestors()

method), 530

ancestors() (pynlpl.formats.folia.CoreferenceLayer method), 659 ancestors() (pynlpl.formats.folia.CoreferenceLink method), 765 ancestors() (pynlpl.formats.folia.Correction method), 945 ancestors() (pynlpl.formats.folia.Current method), 954 ancestors() (pynlpl.formats.folia.Definition method), 115 (pynlpl.formats.folia.DependenciesLayer ancestors() method), 671 ancestors() (pynlpl.formats.folia.Dependency method), ancestors() (pynlpl.formats.folia.DependencyDependent method), 777 ancestors() (pynlpl.formats.folia.Description method), ancestors() (pynlpl.formats.folia.Division method), 128 (pynlpl.formats.folia.DomainAnnotation ancestors() method), 430 ancestors() (pynlpl.formats.folia.EnddatetimeFeature method), 908 ancestors() (pynlpl.formats.folia.EntitiesLayer method), ancestors() (pynlpl.formats.folia.Entity method), 554 ancestors() (pynlpl.formats.folia.Entry method), 140 ancestors() (pynlpl.formats.folia.AbstractAnnotationLayer ancestors() (pynlpl.formats.folia.ErrorDetection method), ancestors() (pynlpl.formats.folia.Event method), 154 ancestors() (pynlpl.formats.folia.Example method), 166 ancestors() (pynlpl.formats.folia.Feature method), 864 ancestors() (pynlpl.formats.folia.Figure method), 180 ancestors() (pynlpl.formats.folia.Head method), 204 ancestors() (pynlpl.formats.folia.Headspan method), 789 ancestors() (pynlpl.formats.folia.LangAnnotation method), 452 (pynlpl.formats.folia.LemmaAnnotation ancestors() method), 463 ancestors() (pynlpl.formats.folia.Linebreak method), 217 ancestors() (pynlpl.formats.folia.List method), 229 ancestors() (pynlpl.formats.folia.ListItem method), 242 ancestors() (pynlpl.formats.folia.Metric method), 1044 ancestors() (pynlpl.formats.folia.New method), 977 ancestors() (pynlpl.formats.folia.Note method), 255 ancestors() (pynlpl.formats.folia.Observation method), 565 ancestors() (pynlpl.formats.folia.ObservationLayer method), 695 ancestors() (pynlpl.formats.folia.Original method), 988 ancestors() (pynlpl.formats.folia.Paragraph method), 269 ancestors() (pynlpl.formats.folia.Part method), 282 ancestors() (pynlpl.formats.folia.PhonContent method), ancestors() (pynlpl.formats.folia.PosAnnotation method), 441

- ancestors() (pynlpl.formats.folia.Predicate method), 577 ancestors() (pynlpl.formats.folia.Quote method), 294 ancestors() (pynlpl.formats.folia.Reference method), 308 ancestors() (pynlpl.formats.folia.Row method), 320 ancestors() (pynlpl.formats.folia.SemanticRole method), ancestors() (pynlpl.formats.folia.SemanticRolesLayer method), 742 (pynlpl.formats.folia.SenseAnnotation ancestors() method), 474 ancestors() (pynlpl.formats.folia.Sentence method), 334 ancestors() (pynlpl.formats.folia.Sentiment method), 589 (pynlpl.formats.folia.SentimentLayer ancestors() method), 707 ancestors() (pynlpl.formats.folia.Statement method), 600 (pynlpl.formats.folia.StatementLayer ancestors() method), 718 ancestors() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 ancestors() (pynlpl.formats.folia.Suggestion method), ancestors() (pynlpl.formats.folia.SynsetFeature method), ancestors() (pynlpl.formats.folia.SyntacticUnit method), ancestors() (pynlpl.formats.folia.SyntaxLayer method), ancestors() (pynlpl.formats.folia.Table method), 348 ancestors() (pynlpl.formats.folia.TableHead method), 374 ancestors() (pynlpl.formats.folia.Term method), 361 ancestors() (pynlpl.formats.folia.Text method), 387 ancestors() (pynlpl.formats.folia.TextContent method), ancestors() (pynlpl.formats.folia.TextMarkupCorrection method), 841 (pynlpl.formats.folia.TextMarkupError ancestors() method), 852 (pynlpl.formats.folia.TextMarkupGap ancestors() method), 809 (pynlpl.formats.folia.TextMarkupString ancestors() method), 820 (pynlpl.formats.folia.TextMarkupStyle ancestors() method), 831 ancestors() (pynlpl.formats.folia.TimeSegment method), ancestors() (pynlpl.formats.folia.TimingLayer method), 754 ancestors() (pynlpl.formats.folia.Whitespace method), ancestors() (pynlpl.formats.folia.Word method), 414 annotation() (pynlpl.formats.folia.AbstractAnnotationLayer method), 76 annotation() (pynlpl.formats.folia.Row method), 321 annotation() (pynlpl.formats.folia.AbstractSpanAnnotation annotation() (pynlpl.formats.folia.SemanticRole method), method), 53
- annotation() (pynlpl.formats.folia.AbstractStructureElement method), 39 annotation() (pynlpl.formats.folia.AllowTokenAnnotation method), 48 annotation() (pynlpl.formats.folia.Alternative method), 920 annotation() (pynlpl.formats.folia.Cell method), 102 annotation() (pynlpl.formats.folia.Chunk method), 519 annotation() (pynlpl.formats.folia.ChunkingLayer method), 648 annotation() (pynlpl.formats.folia.CoreferenceChain method), 531 (pynlpl.formats.folia.CoreferenceLayer annotation() method), 660 annotation() (pynlpl.formats.folia.CoreferenceLink method), 765 annotation() (pynlpl.formats.folia.Definition method), 115 (pynlpl.formats.folia.DependenciesLayer annotation() method), 671 annotation() (pynlpl.formats.folia.Dependency method), 542 annotation() (pynlpl.formats.folia.DependencyDependent method), 777 annotation() (pynlpl.formats.folia.Division method), 128 annotation() (pynlpl.formats.folia.EntitiesLayer method), 683 annotation() (pynlpl.formats.folia.Entity method), 554 annotation() (pynlpl.formats.folia.Entry method), 141 annotation() (pynlpl.formats.folia.Event method), 154 annotation() (pynlpl.formats.folia.Example method), 167 annotation() (pynlpl.formats.folia.Figure method), 180 annotation() (pynlpl.formats.folia.Head method), 204 annotation() (pynlpl.formats.folia.Headspan method), 789 annotation() (pynlpl.formats.folia.Linebreak method), 217 annotation() (pynlpl.formats.folia.List method), 230 annotation() (pynlpl.formats.folia.ListItem method), 243 annotation() (pynlpl.formats.folia.Note method), 256 annotation() (pynlpl.formats.folia.Observation method), 566 annotation() (pynlpl.formats.folia.ObservationLayer method), 695 annotation() (pynlpl.formats.folia.Paragraph method), 269 annotation() (pynlpl.formats.folia.Part method), 282 annotation() (pynlpl.formats.folia.Predicate method), 577 annotation() (pynlpl.formats.folia.Quote method), 295 annotation() (pynlpl.formats.folia.Reference method),

624

$annotation () \\  (pynlpl.formats.folia. Semantic Roles Layer$	method), 777
method), 742	$annotations () \ (pynlpl. formats. folia. Division \ method), \ 128$
annotation() (pynlpl.formats.folia.Sentence method), 335	annotations() (pynlpl.formats.folia.EntitiesLayer
annotation() (pynlpl.formats.folia.Sentiment method),	method), 683
589	annotations() (pynlpl.formats.folia.Entity method), 554
annotation() (pynlpl.formats.folia.SentimentLayer	annotations() (pynlpl.formats.folia.Entry method), 141
method), 707	annotations() (pynlpl.formats.folia.Event method), 154
annotation() (pynlpl.formats.folia.Statement method), 600	annotations() (pynlpl.formats.folia.Example method), 167
annotation() (pynlpl.formats.folia.StatementLayer method), 718	annotations() (pynlpl.formats.folia.Figure method), 180 annotations() (pynlpl.formats.folia.Head method), 204
annotation() (pynlpl.formats.folia.SyntacticUnit method),	annotations() (pynlpl.formats.folia.Headspan method),
612	789
annotation() (pynlpl.formats.folia.SyntaxLayer method), 730	annotations() (pynlpl.formats.folia.Linebreak method), 217
annotation() (pynlpl.formats.folia.Table method), 348	annotations() (pynlpl.formats.folia.List method), 230
annotation() (pynlpl.formats.folia.TableHead method),	annotations() (pynlpl.formats.folia.ListItem method), 243
374	annotations() (pynlpl.formats.folia.Note method), 256
annotation() (pynlpl.formats.folia.Term method), 361 annotation() (pynlpl.formats.folia.Text method), 387	annotations() (pynlpl.formats.folia.Observation method), 566
annotation() (pynlpl.formats.folia.TimeSegment method),	annotations() (pynlpl.formats.folia.ObservationLayer
635	method), 695
annotation() (pynlpl.formats.folia.TimingLayer method), 754	annotations() (pynlpl.formats.folia.Paragraph method), 269
annotation() (pynlpl.formats.folia.Whitespace method),	annotations() (pynlpl.formats.folia.Part method), 282
400	annotations() (pynlpl.formats.folia.Predicate method),
annotation() (pynlpl.formats.folia.Word method), 415	577
annotations() (pynlpl.formats.folia.AbstractAnnotationLayer	eannotations() (pynlpl.formats.folia.Quote method), 295
method), 76	annotations() (pynlpl.formats.folia.Reference method),
$annotations () \ (pynlpl. formats. folia. Abstract Span Annotation \ (pynlpl. formats. folia.) \ (pynlpl. format$	308
method), 53	annotations() (pynlpl.formats.folia.Row method), 321
annotations ()  (pynlpl. formats. folia. Abstract Structure Element annotations)  (pynlpl. formats. folia. Abstract Element annotations	
method), 39	method), 624
annotations() (pynlpl.formats.folia.AllowTokenAnnotation method), 49	annotations() (pynlpl.formats.folia.SemanticRolesLayer method), 742
annotations() (pynlpl.formats.folia.Alternative method),	
921	annotations() (pynlpl.formats.folia.Sentence method), 335
annotations() (pynlpl.formats.folia.Cell method), 102	annotations ()  (pynlpl.formats.folia. Sentence  method),
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer	annotations() (pynlpl.formats.folia.Sentence method), 335 annotations() (pynlpl.formats.folia.Sentiment method), 589 annotations() (pynlpl.formats.folia.SentimentLayer
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer	annotations() (pynlpl.formats.folia.Sentence method), 335 annotations() (pynlpl.formats.folia.Sentiment method), 589 annotations() (pynlpl.formats.folia.SentimentLayer
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer method), 660	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer method), 718
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer method), 660 annotations() (pynlpl.formats.folia.CoreferenceLink	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer method), 718  annotations() (pynlpl.formats.folia.SyntacticUnit
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer method), 660 annotations() (pynlpl.formats.folia.CoreferenceLink method), 765 annotations() (pynlpl.formats.folia.Definition method),	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer method), 718  annotations() (pynlpl.formats.folia.SyntacticUnit method), 612  annotations() (pynlpl.formats.folia.SyntaxLayer method),
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer method), 660 annotations() (pynlpl.formats.folia.CoreferenceLink method), 765	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer method), 718  annotations() (pynlpl.formats.folia.SyntacticUnit method), 612
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer method), 660 annotations() (pynlpl.formats.folia.CoreferenceLink method), 765 annotations() (pynlpl.formats.folia.Definition method), 115	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer method), 718  annotations() (pynlpl.formats.folia.SyntacticUnit method), 612  annotations() (pynlpl.formats.folia.SyntaxLayer method), 730
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer method), 660 annotations() (pynlpl.formats.folia.CoreferenceLink method), 765 annotations() (pynlpl.formats.folia.Definition method), 115 annotations() (pynlpl.formats.folia.DependenciesLayer	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer method), 718  annotations() (pynlpl.formats.folia.SyntacticUnit method), 612  annotations() (pynlpl.formats.folia.SyntaxLayer method), 730  annotations() (pynlpl.formats.folia.Table method), 349
annotations() (pynlpl.formats.folia.Cell method), 102 annotations() (pynlpl.formats.folia.Chunk method), 519 annotations() (pynlpl.formats.folia.ChunkingLayer method), 648 annotations() (pynlpl.formats.folia.CoreferenceChain method), 531 annotations() (pynlpl.formats.folia.CoreferenceLayer method), 660 annotations() (pynlpl.formats.folia.CoreferenceLink method), 765 annotations() (pynlpl.formats.folia.Definition method), 115 annotations() (pynlpl.formats.folia.DependenciesLayer method), 671	annotations() (pynlpl.formats.folia.Sentence method), 335  annotations() (pynlpl.formats.folia.Sentiment method), 589  annotations() (pynlpl.formats.folia.SentimentLayer method), 707  annotations() (pynlpl.formats.folia.Statement method), 601  annotations() (pynlpl.formats.folia.StatementLayer method), 718  annotations() (pynlpl.formats.folia.SyntacticUnit method), 612  annotations() (pynlpl.formats.folia.SyntaxLayer method), 730  annotations() (pynlpl.formats.folia.Table method), 349 annotations() (pynlpl.formats.folia.TableHead method),

annotations()	(pynlpl.formats.folia.TimeSegment	attribute), 77	6	
method), 6	36	ANNOTATIONTYPE	(pynlpl.formats.folia.Descrip	tion
annotations()	(pynlpl.formats.folia.TimingLayer	attribute), 10	31	
method), 7		ANNOTATIONTYPE	(pynlpl.formats.folia.Division	at-
	l.formats.folia.Whitespace method),	tribute), 126	17 1	
401	mornaus.rona. wintespace method);	,	(pynlpl.formats.folia.DomainA	nnotation
	formats falia Ward mathad) 415	attribute), 42		imotation
	l.formats.folia.Word method), 415			г.
	E (pynlpl.formats.folia.AbstractAnnota	-		meFeature
attribute), '		attribute), 90		
ANNOTATIONTYP	E (pynlpl.formats.folia.AbstractEleme	nANNOTATIONTYPE	(pynlpl.formats.folia.EntitiesLa	ıyer
attribute), 2	26	attribute), 68	2	
ANNOTATIONTYP	E (pynlpl.formats.folia.AbstractSpanA	n <b>AddMOT</b> ATIONTYPE	(pynlpl.formats.folia.Er	itity
attribute),	52	attribute), 55	3	·
	E (pynlpl.formats.folia.AbstractStructu		(pynlpl.formats.folia.Entry	at-
attribute),		tribute), 139	(pympinormatsirona.2mr)	ut
	E (pynlpl.formats.folia.AbstractTextM		(nunlal formate folio Error Data	ation
		-		Ction
attribute),		attribute), 96		
	E (pynlpl.formats.folia.AbstractToken.		(pynlpl.formats.folia.Event	at-
attribute),		tribute), 152		
ANNOTATIONTYP	E (pynlpl.formats.folia.ActorFeature	ANNOTATIONTYPE	(pynlpl.formats.folia.Example	at-
attribute),	884	tribute), 165		
ANNOTATIONTYP:		,	(pynlpl.formats.folia.Feature	at-
attribute),	4, 1	tribute), 862	( <del>L</del> )	
* * * * * * * * * * * * * * * * * * * *	E (pynlpl.formats.folia.AlignReference	,	(pynlpl.formats.folia.Fig	riiro
				guie
attribute),		attribute), 17		
ANNOTATIONTYP	1. 1		(pynlpl.formats.folia.Gap	at-
attribute),		tribute), 191		
ANNOTATIONTYP	E (pynlpl.formats.folia.AlternativeLay	eANNOTATIONTYPE	(pynlpl.formats.folia.Head	at-
attribute), 9	931	tribute), 202		
ANNOTATIONTYP	E (pynlpl.formats.folia.Begindatetimel	FAINMOTATIONTYPE	(pynlpl.formats.folia.Headspar	at-
attribute),		tribute), 788	17 1	
ANNOTATIONTYP		,	(pynlpl.formats.folia.LangAnn	ntation
tribute), 10	4.5 1	attribute), 45		Julion
ANNOTATIONTYP				
	45 1		(pynlpl.formats.folia.LemmaA	nnotation
attribute),		attribute), 46		
	E (pynlpl.formats.folia.ChunkingLaye		(pynlpl.formats.folia.Linebreal	at-
attribute),		tribute), 215		
ANNOTATIONTYP	E (pynlpl.formats.folia.CoreferenceCh	a <b>A</b> aNNOTATIONTYPE	(pynlpl.formats.folia.List	at-
attribute),		tribute), 228		
	E (pynlpl.formats.folia.CoreferenceLa		(pynlpl.formats.folia.ListItem	at-
attribute),	=	tribute), 241	(P)peee	
* * * * * * * * * * * * * * * * * * * *	E (pynlpl.formats.folia.CoreferenceLir	,	(nynlni formata folia Ma	trio
	** *		(pynlpl.formats.folia.Me	cuic
attribute),		attribute), 10		
ANNOTATIONTYP	1. 1	ANNOTATIONTYPE	(pynlpl.formats.folia.New	at-
attribute),		tribute), 975		
ANNOTATIONTYP	E (pynlpl.formats.folia.Current at-	ANNOTATIONTYPE	(pynlpl.formats.folia.Note	at-
tribute), 95	53	tribute), 254		
ANNOTATIONTYP	E (pynlpl.formats.folia.Definition at-	ANNOTATIONTYPE	(pynlpl.formats.folia.Observa	tion
tribute), 11	= : =	attribute), 56	4. 1	
	E (pynlpl.formats.folia.DependenciesL	· · · · · · · · · · · · · · · · · · ·		onI aver
		-		onLayer
attribute), (		attribute), 69		.4
	E (pynlpl.formats.folia.Dependency		(pynlpl.formats.folia.Original	at-
attribute),		tribute), 986		
ANNOTATIONTYP	E (pynlpl formats folia Dependency De	DANNINGTATIONTYPE	(pynlpl formats folia Paragraph	at-

tuibuta) 267
tribute), 267 attribute), 851  ANNOTATIONITY DE (nymbol formats folio Port, et ANNOTATIONITY DE (nymbol formats folio Toy t Morly n Con)
ANNOTATIONTYPE (pynlpl.formats.folia.Part at- ANNOTATIONTYPE (pynlpl.formats.folia.TextMarkupGap
tribute), 280 attribute), 808
ANNOTATIONTYPE (pynlpl.formats.folia.PhonContent ANNOTATIONTYPE (pynlpl.formats.folia.TextMarkupString
attribute), 505 attribute), 819
$ANNOTATION TYPE \ (pynlpl. formats. folia. Pos Annotation \ ANNOTATION TYPE \ (pynlpl. formats. folia. TextMarkup Style and the folial position of the folial $
attribute), 439 attribute), 829
ANNOTATIONTYPE (pynlpl.formats.folia.Predicate at- ANNOTATIONTYPE (pynlpl.formats.folia.TimeSegment
tribute), 576 attribute), 634
ANNOTATIONTYPE (pynlpl.formats.folia.Quote ANNOTATIONTYPE (pynlpl.formats.folia.TimingLayer
attribute), 293 attribute), 752
ANNOTATIONTYPE (pynlpl.formats.folia.Reference at- ANNOTATIONTYPE (pynlpl.formats.folia.Whitespace
tribute), 306 attribute), 399
ANNOTATIONTYPE (pynlpl.formats.folia.Row at ANNOTATIONTYPE (pynlpl.formats.folia.Word at-
tribute), 319 tribute), 412
attribute), 622 append() (pynlpl.datatypes.PriorityQueue method), 6
ANNOTATIONTYPE (pynlpl.formats.folia.SemanticRolesLappernd() (pynlpl.datatypes.Tree method), 6
attribute), 740 append() (pynlpl.datatypes.Trie method), 6
ANNOTATIONTYPE (pynlpl.formats.folia.SenseAnnotatioappend() (pynlpl.evaluation.ClassEvaluation method), 9
attribute), 472 append() (pynlpl.evaluation.ExperimentPool method), 10
ANNOTATIONTYPE (pynlpl.formats.folia.Sentence at append() (pynlpl.formats.folia.AbstractAnnotationLayer
tribute), 332 method), 76
ANNOTATIONTYPE (pynlpl.formats.folia.Sentiment at- append() (pynlpl.formats.folia.AbstractElement method),
tribute), 587 27
ANNOTATIONTYPE (pynlpl.formats.folia.SentimentLayerappend() (pynlpl.formats.folia.AbstractSpanAnnotation
attribute), 705 method), 53
ANNOTATIONTYPE (pynlpl.formats.folia.Statement at- append() (pynlpl.formats.folia.AbstractStructureElement
tribute), 599 method), 39
ANNOTATIONTYPE (pynlpl.formats.folia.StatementLayerappend() (pynlpl.formats.folia.AbstractTextMarkup
attribute), 717 method), 87
ANNOTATIONTYPE (pynlpl.formats.folia.SubjectivityAnnapptiod() (pynlpl.formats.folia.AbstractTokenAnnotation
attribute), 483 method), 64
ANNOTATIONTYPE (pynlpl.formats.folia.Suggestion append() (pynlpl.formats.folia.ActorFeature method),
attribute), 997 886
ANNOTATIONTYPE (pynlpl.formats.folia.SynsetFeature append() (pynlpl.formats.folia.Alignment method), 1010
attribute), 873 append() (pynlpl.formats.folia.AlignReference method),
ANNOTATIONTYPE (pynlpl.formats.folia.SyntacticUnit 1021
attribute), 611 append() (pynlpl.formats.folia.Alternative method), 921
ANNOTATIONTYPE (pynlpl.formats.folia.SyntaxLayer append() (pynlpl.formats.folia.AlternativeLayers
attribute), 729 method), 932
ANNOTATIONTYPE (pynlpl.formats.folia.Table at- append() (pynlpl.formats.folia.BegindatetimeFeature
tribute), 347 method), 897
ANNOTATIONTYPE (pynlpl.formats.folia.TableHead append() (pynlpl.formats.folia.Cell method), 103
attribute), 373 append() (pynlpl.formats.folia.Chunk method), 519
tribute), 360 648
ANNOTATIONTYPE (pynlpl.formats.folia.Text at append() (pynlpl.formats.folia.CoreferenceChain
tribute), 386 method), 531
ANNOTATIONTYPE (pynlpl.formats.folia.TextContent append() (pynlpl.formats.folia.CoreferenceLayer
attribute), 495 method), 660
ANNOTATIONTYPE (pynlpl.formats.folia.TextMarkupCorappional() (pynlpl.formats.folia.CoreferenceLink method),
attribute), 840 766
ANNOTATIONTYPE (pynlpl.formats.folia.TextMarkupErrorppend() (pynlpl.formats.folia.Correction method), 945

- append() (pynlpl.formats.folia.Current method), 955 append() (pynlpl.formats.folia.Definition method), 116 append() (pynlpl.formats.folia.DependenciesLayer method), 672 append() (pynlpl.formats.folia.Dependency method), 543 (pynlpl.formats.folia.DependencyDependent append() method), 777 append() (pynlpl.formats.folia.Description method), 1033 append() (pynlpl.formats.folia.Division method), 129 append() (pynlpl.formats.folia.Document method), 17 append() (pynlpl.formats.folia.DomainAnnotation method), 430 (pynlpl.formats.folia.EnddatetimeFeature append() method), 908 append() (pynlpl.formats.folia.EntitiesLayer method), append() (pynlpl.formats.folia.Entity method), 554 append() (pynlpl.formats.folia.Entry method), 141 append() (pynlpl.formats.folia.ErrorDetection method), append() (pynlpl.formats.folia.Event method), 155 append() (pynlpl.formats.folia.Example method), 167 append() (pynlpl.formats.folia.Feature method), 864 append() (pynlpl.formats.folia.Figure method), 181 append() (pynlpl.formats.folia.Gap method), 192 append() (pynlpl.formats.folia.Head method), 205 append() (pynlpl.formats.folia.Headspan method), 789 append() (pynlpl.formats.folia.LangAnnotation method), (pynlpl.formats.folia.LemmaAnnotation append() method), 463 append() (pynlpl.formats.folia.Linebreak method), 218 append() (pynlpl.formats.folia.List method), 230 append() (pynlpl.formats.folia.ListItem method), 243 append() (pynlpl.formats.folia.Metric method), 1044 append() (pynlpl.formats.folia.New method), 977 append() (pynlpl.formats.folia.Note method), 256 append() (pynlpl.formats.folia.Observation method), 566 (pynlpl.formats.folia.ObservationLayer append() method), 695 append() (pynlpl.formats.folia.Original method), 988 append() (pynlpl.formats.folia.Paragraph method), 270 append() (pynlpl.formats.folia.Part method), 283 append() (pynlpl.formats.folia.PhonContent method), 507 append() (pynlpl.formats.folia.PosAnnotation method), 441 append() (pynlpl.formats.folia.Predicate method), 577 append() (pynlpl.formats.folia.Quote method), 295 append() (pynlpl.formats.folia.Reference method), 309 append() (pynlpl.formats.folia.Row method), 321 append() (pynlpl.formats.folia.SemanticRole method), (pynlpl.formats.folia.SemanticRolesLayer append() method), 742
- append() (pynlpl.formats.folia.SenseAnnotation method), append() (pynlpl.formats.folia.Sentence method), 335 append() (pynlpl.formats.folia.Sentiment method), 589 append() (pynlpl.formats.folia.SentimentLayer method), append() (pynlpl.formats.folia.Statement method), 601 append() (pynlpl.formats.folia.StatementLayer method), append() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 append() (pynlpl.formats.folia.Suggestion method), 999 append() (pynlpl.formats.folia.SynsetFeature method), 875 append() (pynlpl.formats.folia.SyntacticUnit method), append() (pynlpl.formats.folia.SyntaxLayer method), 730 append() (pynlpl.formats.folia.Table method), 349 append() (pynlpl.formats.folia.TableHead method), 375 append() (pynlpl.formats.folia.Term method), 362 append() (pynlpl.formats.folia.Text method), 388 append() (pynlpl.formats.folia.TextContent method), 496 (pynlpl.formats.folia.TextMarkupCorrection append() method), 842 (pynlpl.formats.folia.TextMarkupError append() method), 852 append() (pynlpl.formats.folia.TextMarkupGap method), (pynlpl.formats.folia.TextMarkupString append() method), 820 append() (pynlpl.formats.folia.TextMarkupStyle method), 831 append() (pynlpl.formats.folia.TimeSegment method), append() (pynlpl.formats.folia.TimingLayer method), 754 append() (pynlpl.formats.folia.Whitespace method), 401 append() (pynlpl.formats.folia.Word method), 415 append() (pynlpl.lm.lm.SimpleLanguageModel method), 1057 append() (pynlpl.statistics.FrequencyList method), 1063 ARPALanguageModel (class in pynlpl.lm.lm), 1057 ARPALanguageModel.NgramsProbs (class in pvnlpl.lm.lm), 1057 auc() (in module pynlpl.evaluation), 10 auc() (pynlpl.evaluation.ClassEvaluation method), 9 AUTH (pynlpl.formats.folia.AbstractAnnotationLaver attribute), 74 AUTH (pynlpl.formats.folia.AbstractElement attribute), AUTH (pynlpl.formats.folia.AbstractSpanAnnotation attribute), 52

(pynlpl.formats.folia.AbstractStructureElement

(pynlpl.formats.folia.AbstractTextMarkup at-

Index 1093

**AUTH** 

**AUTH** 

attribute), 37

tribute), 86	AUTH (pynlpl.formats.folia.ListItem attribute), 241
AUTH (pynlpl.formats.folia.AbstractTokenAnnotation	AUTH (pynlpl.formats.folia.Metric attribute), 1042
attribute), 63	AUTH (pynlpl.formats.folia.New attribute), 975
AUTH (pynlpl.formats.folia.ActorFeature attribute), 885	AUTH (pynlpl.formats.folia.Note attribute), 254
AUTH (pynlpl.formats.folia.Alignment attribute), 1009	AUTH (pynlpl.formats.folia.Observation attribute), 564
AUTH (pynlpl.formats.folia.AlignReference attribute), 1020	AUTH (pynlpl.formats.folia.ObservationLayer attribute), 693
AUTH (pynlpl.formats.folia.Alternative attribute), 919	AUTH (pynlpl.formats.folia.Original attribute), 986
AUTH (pynlpl.formats.folia.AlternativeLayers attribute),	AUTH (pynlpl.formats.folia.Paragraph attribute), 267
931	AUTH (pynlpl.formats.folia.Part attribute), 280
AUTH (pynlpl.formats.folia.BegindatetimeFeature	AUTH (pynlpl.formats.folia.PhonContent attribute), 505
attribute), 896	AUTH (pynlpl.formats.folia.PosAnnotation attribute),
AUTH (pynlpl.formats.folia.Cell attribute), 100	439
AUTH (pynlpl.formats.folia.Chunk attribute), 518	AUTH (pynlpl.formats.folia.Predicate attribute), 576
AUTH (pynlpl.formats.folia.ChunkingLayer attribute),	AUTH (pynlpl.formats.folia.Quote attribute), 293
646	AUTH (pynlpl.formats.folia.Reference attribute), 306
AUTH (pynlpl.formats.folia.CoreferenceChain attribute),	AUTH (pynlpl.formats.folia.Row attribute), 319
529	AUTH (pynlpl.formats.folia.SemanticRole attribute), 622
AUTH (pynlpl.formats.folia.CoreferenceLayer attribute), 658	AUTH (pynlpl.formats.folia.SemanticRolesLayer attribute), 740
AUTH (pynlpl.formats.folia.CoreferenceLink attribute), 764	AUTH (pynlpl.formats.folia.SenseAnnotation attribute), 472
AUTH (pynlpl.formats.folia.Correction attribute), 944	AUTH (pynlpl.formats.folia.Sentence attribute), 332
AUTH (pynlpl.formats.folia.Current attribute), 953	AUTH (pynlpl.formats.folia.Sentiment attribute), 587
AUTH (pynlpl.formats.folia.Definition attribute), 113	AUTH (pynlpl.formats.folia.SentimentLayer attribute),
AUTH (pynlpl.formats.folia.DependenciesLayer attribute), 670	705 AUTH (pynlpl.formats.folia.Statement attribute), 599
AUTH (pynlpl.formats.folia.Dependency attribute), 541	AUTH (pynlpl.formats.folia.StatementLayer attribute),
AUTH (pynlpl.formats.folia.DependencyDependent at-	717
tribute), 776	AUTH (pynlpl.formats.folia.SubjectivityAnnotation at-
AUTH (pynlpl.formats.folia.Description attribute), 1031	tribute), 483
AUTH (pynlpl.formats.folia.Division attribute), 126	AUTH (pynlpl.formats.folia.Suggestion attribute), 997
AUTH (pynlpl.formats.folia.DomainAnnotation at-	AUTH (pynlpl.formats.folia.SynsetFeature attribute), 873
tribute), 428	AUTH (pynlpl.formats.folia.SyntacticUnit attribute), 611
AUTH (pynlpl.formats.folia.EnddatetimeFeature at-	AUTH (pynlpl.formats.folia.SyntaxLayer attribute), 729
tribute), 907	AUTH (pynlpl.formats.folia.Table attribute), 347
AUTH (pynlpl.formats.folia.EntitiesLayer attribute), 682 AUTH (pynlpl.formats.folia.Entity attribute), 553	AUTH (pynlpl.formats.folia.TableHead attribute), 373 AUTH (pynlpl.formats.folia.Term attribute), 360
AUTH (pynlpl.formats.folia.Entry attribute), 333	AUTH (pynlpl.formats.folia.Text attribute), 386
AUTH (pynlpl.formats.folia.ErrorDetection attribute),	AUTH (pynlpl.formats.folia.TextContent attribute), 495
964	AUTH (pynlpl.formats.folia.TextMarkupCorrection at-
AUTH (pynlpl.formats.folia.Event attribute), 152	tribute), 840
AUTH (pynlpl.formats.folia.Example attribute), 165	AUTH (pynlpl.formats.folia.TextMarkupError attribute),
AUTH (pynlpl.formats.folia.Feature attribute), 862	851
AUTH (pynlpl.formats.folia.Figure attribute), 178	AUTH (pynlpl.formats.folia.TextMarkupGap attribute),
AUTH (pynlpl.formats.folia.Gap attribute), 191	808
AUTH (pynlpl.formats.folia.Head attribute), 202	AUTH (pynlpl.formats.folia.TextMarkupString attribute),
AUTH (pynlpl.formats.folia.Headspan attribute), 788	819
AUTH (pynlpl.formats.folia.LangAnnotation attribute), 450	AUTH (pynlpl.formats.folia.TextMarkupStyle attribute), 829
AUTH (pynlpl.formats.folia.LemmaAnnotation at-	AUTH (pynlpl.formats.folia.TimeSegment attribute), 634
tribute), 461	AUTH (pynlpl.formats.folia.TimingLayer attribute), 752
AUTH (pynlpl.formats.folia.Linebreak attribute), 215	AUTH (pynlpl.formats.folia.Whitespace attribute), 399
AUTH (pynlpl.formats.folia.List attribute), 228	AUTH (pynlpl.formats.folia.Word attribute), 412

- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AbstractAnno**AdtiT60L\_G/E**NERATE\_ID (pynlpl.formats.folia.EnddatetimeFeature attribute), 74 attribute), 907
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AbstractElem**Au**tTO\_GENERATE\_ID (pynlpl.formats.folia.EntitiesLayer attribute), 26 attribute), 682
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AbstractSpanAb/Tctatf@ENERATE\_ID (pynlpl.formats.folia.Entity atattribute), 52 tribute), 553
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AbstractStrucAlt/ETDe\_GENERATE\_ID (pynlpl.formats.folia.Entry atattribute), 37 tribute), 139
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AbstractTextMarttop\_GENERATE\_ID (pynlpl.formats.folia.ErrorDetection attribute), 86 attribute), 964
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AbstractTokerAA/TrQtatiENERATE\_ID (pynlpl.formats.folia.Event atattribute), 63 tribute), 152
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.ActorFeature AUTO\_GENERATE\_ID (pynlpl.formats.folia.Example attribute), 885 attribute), 165
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Alignment attribute), 1009 AUTO\_GENERATE\_ID (pynlpl.formats.folia.Feature attribute), 862
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AlignReferen&UTO\_GENERATE\_ID (pynlpl.formats.folia.Figure atattribute), 1020 tribute), 178
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Alternative AUTO\_GENERATE\_ID attribute), 919 (pynlpl.formats.folia.Gap attribute), 191
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.AlternativeLanderTO\_GENERATE\_ID (pynlpl.formats.folia.Head atattribute), 931 tribute), 202
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Begindatetim AETOrGENERATE\_ID (pynlpl.formats.folia.Headspan attribute), 896 attribute), 788
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Cell AUTO\_GENERATE\_ID (pynlpl.formats.folia.LangAnnotation attribute), 100
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Chunk attribute), 518 AUTO\_GENERATE\_ID (pynlpl.formats.folia.LemmaAnnotation attribute), 461
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.ChunkingLayAUTO\_GENERATE\_ID (pynlpl.formats.folia.Linebreak attribute), 646 attribute), 215
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.CoreferenceCALITO\_GENERATE\_ID attribute), 529 (pynlpl.formats.folia.List
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.CoreferenceLAYEFO\_GENERATE\_ID (pynlpl.formats.folia.ListItem attribute), 658 attribute), 241
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.CoreferenceLANTO\_GENERATE\_ID (pynlpl.formats.folia.Metric atattribute), 764 tribute), 1042
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Correction AUTO\_GENERATE\_ID (pynlpl.formats.folia.New atattribute), 944 tribute), 975
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Current attribute), 953 (pynlpl.formats.folia.Note attribute), 254
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Definition AUTO\_GENERATE\_ID (pynlpl.formats.folia.Observation attribute), 113 attribute), 564
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Dependencies AUJO\_GENERATE\_ID (pynlpl.formats.folia.ObservationLayer attribute), 670 attribute), 693
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Dependency AUTO\_GENERATE\_ID attribute), 541 (pynlpl.formats.folia.Original attribute), 986
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.DependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDepende
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Description AUTO\_GENERATE\_ID attribute), 1031 (pynlpl.formats.folia.Part
- AUTO\_GENERATE\_ID (pynlpl.formats.folia.Division AUTO\_GENERATE\_ID (pynlpl.formats.folia.PhonContent attribute), 126 attribute), 505
- $AUTO\_GENERATE\_ID\ (pynlpl.formats.folia.DomainAnno \textbf{Autibito} \_GENERATE\_ID\ (pynlpl.formats.folia.PosAnnotation\ attribute), 428$

AUTO_GENERATE_ID (pynlpl.formats.folia.Predicate attribute), 576	AUTO_GENERATE_ID (pynlpl.formats.folia.TimeSegment attribute), 634
AUTO_GENERATE_ID (pynlpl.formats.folia.Quote at-	AUTO_GENERATE_ID (pynlpl.formats.folia.TimingLayer
tribute), 293	attribute), 752
AUTO_GENERATE_ID (pynlpl.formats.folia.Reference attribute), 306	AUTO_GENERATE_ID (pynlpl.formats.folia.Whitespace attribute), 399
AUTO_GENERATE_ID (pynlpl.formats.folia.Row attribute), 319	AUTO_GENERATE_ID (pynlpl.formats.folia.Word attribute), 412
AUTO_GENERATE_ID (pynlpl.formats.folia.SemanticRo attribute), 623	B
AUTO_GENERATE_ID (pynlpl.formats.folia.SemanticRo	lack over 11 1 11 11 11 12 12
attribute), 740	$\underset{\cdot \cdot }{\text{backoff}()}(pynlpl.lm.lm.ARPALanguageModel.NgramsProbs}$
AUTO_GENERATE_ID (pynlpl.formats.folia.SenseAnnot	
attribute), 472	BeamedBestFirstSearch (class in pynlpl.search), 1060
AUTO_GENERATE_ID (pynlpl.formats.folia.Sentence	BeamSearch (class in pynlpl.search), 1060
attribute), 332	BegindatetimeFeature (class in pynlpl.formats.folia), 893
AUTO_GENERATE_ID (pynlpl.formats.folia.Sentiment	BestFirstSearch (class in pynlpl.search), 1060
attribute), 588	binary_search() (in module pynlpl.search), 1060
AUTO_GENERATE_ID (pynlpl.formats.folia.SentimentLa	MPFandth First Sparch (alogs in punish sparch) 1060
attribute), 705	Breaduir itsisearch (class in pympi.search), 1000
	C
AUTO_GENERATE_ID (pynlpl.formats.folia.Statement	C .
attribute), 599	<pre>calculate_overlap() (in module pynlpl.textprocessors),</pre>
AUTO_GENERATE_ID (pynlpl.formats.folia.StatementLa	yer 1066
attribute), 717	_caption() (pynlpl.formats.folia.Figure method), 181
AUTO_GENERATE_ID (pynlpl.formats.folia.Subjectivity.	Applications in pynlpl formats folia) 98
attribute), 483	Chunk (class in pynlpl.formats.folia), 515
AUTO_GENERATE_ID (pynlpl.formats.folia.Suggestion	
attribute), 997	ChunkingLayer (class in pynlpl.formats.folia), 644
AUTO GENERATE ID (nynlnl formats folia SynsetFeatu	ClassEvaluation (class in pynlpl.evaluation), 9
AUTO_GENERATE_ID (pynlpl.formats.folia.SynsetFeatu attribute), 873	close() (pynlpl.formats.taggerdata.Taggerdata method),
AUTO_GENERATE_ID (pynlpl.formats.folia.SyntacticUn attribute), 611	<sup>1</sup> communicates() (pynlpl.statistics.MarkovChain method),
attribute), 011	1063
AUTO_GENERATE_ID (pynlpl.formats.folia.SyntaxLayer	compute() (pynlpl.evaluation.ClassEvaluation method), 9
attribute), 729	compute() (pynlpl.evaluation.OrdinalEvaluation method),
AUTO_GENERATE_ID (pynlpl.formats.folia.Table at-	10
tribute), 347	ConfusionMatrix (class in pynlpl.evaluation), 10
AUTO_GENERATE_ID (pynlpl.formats.folia.TableHead	confusionmatrix() (pynlpl.evaluation.ClassEvaluation
attribute), 373	method), 9
AUTO_GENERATE_ID (pynlpl.formats.folia.Term at-	content() (pynlpl.formats.folia.Gap method), 192
tribute), 360	
AUTO_GENERATE_ID (pynlpl.formats.folia.Text at-	context() (pynlpl.formats.folia.AbstractAnnotationLayer
tribute), 386	method), 76
AUTO_GENERATE_ID (pynlpl.formats.folia.TextContent	context() (pynlpl.formats.folia.AbstractElement method),
attribute), 495	21
ALITO CENEDATE ID (nominal formante folio Tout Mondour	context() (pynlpl.formats.folia.AbstractSpanAnnotation
AUTO_GENERATE_ID (pynlpl.formats.folia.TextMarkup	method), 53
attribute), 840	context() (pynlpl.formats.folia.AbstractStructureElement
AUTO_GENERATE_ID (pynlpl.formats.folia.TextMarkup	Error method), 40
attribute), 851	context() (pynlpl.formats.folia.AbstractTextMarkup
AUTO_GENERATE_ID (pynlpl.formats.folia.TextMarkup	Gap method), 87
attribute), 808	context() (pynlpl.formats.folia.AbstractTokenAnnotation
AUTO_GENERATE_ID (pynlpl.formats.folia.TextMarkup	String method), 64
attribute), 819	method), 64
AUTO_GENERATE_ID (pynlpl.formats.folia.TextMarkup	Ctulo
attribute), 830	000
,,	context() (pynlpl.formats.folia.Alignment method), 1010

context() (pynlpl.formats.folia.AlignReference method), 1021 context() (pynlpl.formats.folia.Alternative method), 921 (pynlpl.formats.folia.AlternativeLayers context() method), 932 (pynlpl.formats.folia.BegindatetimeFeature context() method), 897 context() (pynlpl.formats.folia.Cell method), 103 context() (pynlpl.formats.folia.Chunk method), 519 context() (pynlpl.formats.folia.ChunkingLayer method), (pynlpl.formats.folia.CoreferenceChain context() method), 531 (pynlpl.formats.folia.CoreferenceLayer context() method), 660 context() (pynlpl.formats.folia.CoreferenceLink method), context() (pynlpl.formats.folia.Correction method), 945 context() (pynlpl.formats.folia.Current method), 955 context() (pynlpl.formats.folia.Definition method), 116 context() (pynlpl.formats.folia.DependenciesLayer method), 672 context() (pynlpl.formats.folia.Dependency method), 543 context() (pynlpl.formats.folia.DependencyDependent method), 777 context() (pynlpl.formats.folia.Description method), 1033 context() (pynlpl.formats.folia.Division method), 129 (pynlpl.formats.folia.DomainAnnotation context() method), 430 (pynlpl.formats.folia.EnddatetimeFeature context() method), 908 context() (pynlpl.formats.folia.EntitiesLayer method), 684 context() (pynlpl.formats.folia.Entity method), 554 context() (pynlpl.formats.folia.Entry method), 142 context() (pynlpl.formats.folia.ErrorDetection method), 966 context() (pynlpl.formats.folia.Event method), 155 context() (pynlpl.formats.folia.Example method), 168 context() (pynlpl.formats.folia.Feature method), 864 context() (pynlpl.formats.folia.Figure method), 181 context() (pynlpl.formats.folia.Gap method), 192 context() (pynlpl.formats.folia.Head method), 205 context() (pynlpl.formats.folia.Headspan method), 789 context() (pynlpl.formats.folia.LangAnnotation method), 452 (pynlpl.formats.folia.LemmaAnnotation context() method), 463 context() (pynlpl.formats.folia.Linebreak method), 218 context() (pynlpl.formats.folia.List method), 231 context() (pynlpl.formats.folia.ListItem method), 244 context() (pynlpl.formats.folia.Metric method), 1044

context() (pynlpl.formats.folia.New method), 977

context() (pynlpl.formats.folia.Note method), 257

context() (pynlpl.formats.folia.Observation method), 566 context() (pynlpl.formats.folia.ObservationLayer method), 695 context() (pynlpl.formats.folia.Original method), 988 context() (pynlpl.formats.folia.Paragraph method), 270 context() (pynlpl.formats.folia.Part method), 283 context() (pynlpl.formats.folia.PhonContent method), context() (pynlpl.formats.folia.PosAnnotation method), context() (pynlpl.formats.folia.Predicate method), 577 context() (pynlpl.formats.folia.Quote method), 296 context() (pynlpl.formats.folia.Reference method), 309 context() (pynlpl.formats.folia.Row method), 322 context() (pynlpl.formats.folia.SemanticRole method), 624 context() (pynlpl.formats.folia.SemanticRolesLayer method), 742 context() (pynlpl.formats.folia.SenseAnnotation method), 474 context() (pynlpl.formats.folia.Sentence method), 336 context() (pynlpl.formats.folia.Sentiment method), 589 context() (pynlpl.formats.folia.SentimentLayer method), context() (pynlpl.formats.folia.Statement method), 601 context() (pynlpl.formats.folia.StatementLayer method), 719 (pynlpl.formats.folia.SubjectivityAnnotation context() method), 485 context() (pynlpl.formats.folia.Suggestion method), 999 context() (pynlpl.formats.folia.SynsetFeature method), context() (pynlpl.formats.folia.SyntacticUnit method), context() (pynlpl.formats.folia.SyntaxLayer method), 731 context() (pynlpl.formats.folia.Table method), 349 context() (pynlpl.formats.folia.TableHead method), 375 context() (pynlpl.formats.folia.Term method), 362 context() (pynlpl.formats.folia.Text method), 388 context() (pynlpl.formats.folia.TextContent method), 496 (pynlpl.formats.folia.TextMarkupCorrection context() method), 842 (pynlpl.formats.folia.TextMarkupError context() method), 852 context() (pynlpl.formats.folia.TextMarkupGap method), 810 (pynlpl.formats.folia.TextMarkupString context() method), 820 (pynlpl.formats.folia.TextMarkupStyle context() method), 831 context() (pynlpl.formats.folia.TimeSegment method),

(pynlpl.formats.folia.TimingLayer method),

Index 1097

context()

754

context() (pynlpl.formats.folia.Whitespace method), 401 context() (pynlpl.formats.folia.Word method), 416	copy() (pynlpl.formats.folia.Figure method), 181 copy() (pynlpl.formats.folia.Gap method), 192
copy() (pynlpl.formats.folia.AbstractAnnotationLayer	copy() (pynlpl.formats.folia.Head method), 205
method), 76	copy() (pynlpl.formats.folia.Headspan method), 789
	copy() (pynlpl.formats.folia.LangAnnotation method),
copy() (pynlpl.formats.folia.AbstractElement method),	452
copy() (pynlpl.formats.folia.AbstractSpanAnnotation	copy() (pynlpl.formats.folia.LemmaAnnotation method),
method), 53	463
copy() (pynlpl.formats.folia.AbstractStructureElement	copy() (pynlpl.formats.folia.Linebreak method), 218
method), 40	copy() (pynlpl.formats.folia.List method), 231
copy() (pynlpl.formats.folia.AbstractTextMarkup	copy() (pynlpl.formats.folia.ListItem method), 244
method), 87	copy() (pynlpl.formats.folia.Metric method), 1044
copy() (pynlpl.formats.folia.AbstractTokenAnnotation	copy() (pynlpl.formats.folia.New method), 977
method), 65	copy() (pynlpl.formats.folia.Note method), 257
copy() (pynlpl.formats.folia.ActorFeature method), 886	copy() (pynlpl.formats.folia.Observation method), 566
copy() (pynlpl.formats.folia.Alignment method), 1010	copy() (pynlpl.formats.folia.ObservationLayer method),
copy() (pynlpl.formats.folia.AlignReference method),	695
1021	copy() (pynlpl.formats.folia.Original method), 988
copy() (pynlpl.formats.folia.Alternative method), 921	copy() (pynlpl.formats.folia.Paragraph method), 270
copy() (pynlpl.formats.folia.AlternativeLayers method),	copy() (pynlpl.formats.folia.Part method), 283
932	copy() (pynlpl.formats.folia.PhonContent method), 507
copy() (pynlpl.formats.folia.BegindatetimeFeature	copy() (pynlpl.formats.folia.PosAnnotation method), 441
method), 897	copy() (pynlpl.formats.folia.Predicate method), 578
copy() (pynlpl.formats.folia.Cell method), 103	copy() (pynlpl.formats.folia.Quote method), 296
copy() (pynlpl.formats.folia.Chunk method), 519	copy() (pynlpl.formats.folia.Reference method), 309
copy() (pynlpl.formats.folia.ChunkingLayer method),	copy() (pynlpl.formats.folia.Row method), 322
648	copy() (pynlpl.formats.folia.SemanticRole method), 624
copy() (pynlpl.formats.folia.CoreferenceChain method),	copy() (pynlpl.formats.folia.SemanticRolesLayer method), 742
copy() (pynlpl.formats.folia.CoreferenceLayer method),	copy() (pynlpl.formats.folia.SenseAnnotation method),
660	
	4/4
	474 copy() (pynlpl formats folia Sentence method), 336
copy() (pynlpl.formats.folia.CoreferenceLink method),	copy() (pynlpl.formats.folia.Sentence method), 336
copy() (pynlpl.formats.folia.CoreferenceLink method), 766	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method),
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method),
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method),	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntacticUnit method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature method), 908	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375 copy() (pynlpl.formats.folia.Term method), 362
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature method), 908 copy() (pynlpl.formats.folia.EntitiesLayer method), 684	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375 copy() (pynlpl.formats.folia.Term method), 362 copy() (pynlpl.formats.folia.Text method), 388
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature method), 908 copy() (pynlpl.formats.folia.EntitiesLayer method), 684 copy() (pynlpl.formats.folia.EntitiesLayer method), 554	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375 copy() (pynlpl.formats.folia.Term method), 362 copy() (pynlpl.formats.folia.Text method), 388 copy() (pynlpl.formats.folia.Text method), 496
copy() (pynlpl.formats.folia.CoreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature method), 908 copy() (pynlpl.formats.folia.EntitiesLayer method), 684 copy() (pynlpl.formats.folia.Entity method), 554 copy() (pynlpl.formats.folia.Entity method), 142	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375 copy() (pynlpl.formats.folia.Text method), 388 copy() (pynlpl.formats.folia.Text method), 496 copy() (pynlpl.formats.folia.TextContent method), 496 copy() (pynlpl.formats.folia.TextMarkupCorrection
copy() (pynlpl.formats.folia.CorreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature method), 908 copy() (pynlpl.formats.folia.EntitiesLayer method), 684 copy() (pynlpl.formats.folia.Entity method), 554 copy() (pynlpl.formats.folia.Entry method), 142 copy() (pynlpl.formats.folia.ErrorDetection method), 966	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 875 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375 copy() (pynlpl.formats.folia.Text method), 388 copy() (pynlpl.formats.folia.Text method), 496 copy() (pynlpl.formats.folia.TextContent method), 496 copy() (pynlpl.formats.folia.TextMarkupCorrection method), 842
copy() (pynlpl.formats.folia.CorreferenceLink method), 766 copy() (pynlpl.formats.folia.Current method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature method), 908 copy() (pynlpl.formats.folia.EntitiesLayer method), 684 copy() (pynlpl.formats.folia.Entity method), 554 copy() (pynlpl.formats.folia.Entry method), 142 copy() (pynlpl.formats.folia.ErrorDetection method), 966 copy() (pynlpl.formats.folia.Event method), 155	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 999 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375 copy() (pynlpl.formats.folia.Text method), 388 copy() (pynlpl.formats.folia.Text method), 388 copy() (pynlpl.formats.folia.TextMarkupCorrection method), 842 copy() (pynlpl.formats.folia.TextMarkupError method),
copy() (pynlpl.formats.folia.CorreferenceLink method), 766 copy() (pynlpl.formats.folia.Correction method), 945 copy() (pynlpl.formats.folia.Current method), 955 copy() (pynlpl.formats.folia.Definition method), 116 copy() (pynlpl.formats.folia.DependenciesLayer method), 672 copy() (pynlpl.formats.folia.Dependency method), 543 copy() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.DomainAnnotation method), 430 copy() (pynlpl.formats.folia.EnddatetimeFeature method), 908 copy() (pynlpl.formats.folia.EntitiesLayer method), 684 copy() (pynlpl.formats.folia.Entity method), 554 copy() (pynlpl.formats.folia.Entry method), 142 copy() (pynlpl.formats.folia.ErrorDetection method), 966	copy() (pynlpl.formats.folia.Sentence method), 336 copy() (pynlpl.formats.folia.Sentiment method), 589 copy() (pynlpl.formats.folia.SentimentLayer method), 707 copy() (pynlpl.formats.folia.Statement method), 601 copy() (pynlpl.formats.folia.StatementLayer method), 719 copy() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 copy() (pynlpl.formats.folia.Suggestion method), 875 copy() (pynlpl.formats.folia.SynsetFeature method), 875 copy() (pynlpl.formats.folia.SyntacticUnit method), 613 copy() (pynlpl.formats.folia.SyntaxLayer method), 731 copy() (pynlpl.formats.folia.Table method), 349 copy() (pynlpl.formats.folia.TableHead method), 375 copy() (pynlpl.formats.folia.Text method), 388 copy() (pynlpl.formats.folia.Text method), 496 copy() (pynlpl.formats.folia.TextContent method), 496 copy() (pynlpl.formats.folia.TextMarkupCorrection method), 842

copy() (pynlpl.formats.folia.TextMarkupGap method), copychildren() (pynlpl.formats.folia.DependencyDependent method), 778 copy() (pynlpl.formats.folia.TextMarkupString method), copychildren() (pynlpl.formats.folia.Description method), 1033 copy() (pynlpl.formats.folia.TextMarkupStyle method), copychildren() (pynlpl.formats.folia.Division method), 129 copy() (pynlpl.formats.folia.TimeSegment method), 636 copychildren() (pynlpl.formats.folia.DomainAnnotation copy() (pynlpl.formats.folia.TimingLayer method), 754 method), 430 copy() (pynlpl.formats.folia.Whitespace method), 401 copychildren() (pynlpl.formats.folia.EnddatetimeFeature method), 908 copy() (pynlpl.formats.folia.Word method), 416 copychildren() (pynlpl.formats.folia.AbstractAnnotationLayeopychildren() (pynlpl.formats.folia.EntitiesLayer method), 76 method), 684 (pynlpl.formats.folia.AbstractElement copychildren() (pynlpl.formats.folia.Entity method), 554 copychildren() copychildren() (pynlpl.formats.folia.Entry method), 142 method), 27 copychildren() (pynlpl.formats.folia.AbstractSpanAnnotationopychildren() (pynlpl.formats.folia.ErrorDetection method), 54 method), 966 copychildren() (pynlpl.formats.folia.AbstractStructureElementpychildren() (pynlpl.formats.folia.Event method), 155 copychildren() (pynlpl.formats.folia.Example method), method), 40 copychildren() (pynlpl.formats.folia.AbstractTextMarkup 168 method), 88 (pynlpl.formats.folia.Feature method), copychildren() copychildren() (pynlpl.formats.folia.AbstractTokenAnnotation 864 method), 65 copychildren() (pynlpl.formats.folia.Figure method), 181 copychildren() (pynlpl.formats.folia.Gap method), 192 copychildren() (pynlpl.formats.folia.ActorFeature method), 886 copychildren() (pynlpl.formats.folia.Head method), 205 copychildren() (pynlpl.formats.folia.Headspan method), copychildren() (pynlpl.formats.folia.Alignment method), 1010 789 copychildren() (pynlpl.formats.folia.AlignReference copychildren() (pynlpl.formats.folia.LangAnnotation method), 1021 method), 452 copychildren() (pynlpl.formats.folia.Alternative method), (pynlpl.formats.folia.LemmaAnnotation copychildren() 921 method), 463 copychildren() (pynlpl.formats.folia.AlternativeLayers copychildren() (pynlpl.formats.folia.Linebreak method), method), 933 218 copychildren() (pynlpl.formats.folia.BegindatetimeFeature copychildren() (pynlpl.formats.folia.List method), 231 method), 897 copychildren() (pynlpl.formats.folia.ListItem method), copychildren() (pynlpl.formats.folia.Cell method), 103 244 copychildren() (pynlpl.formats.folia.Chunk method), 520 copychildren() (pynlpl.formats.folia.Metric method). copychildren() (pynlpl.formats.folia.ChunkingLayer 1044 method), 648 copychildren() (pynlpl.formats.folia.New method), 977 (pynlpl.formats.folia.CoreferenceChain copychildren() (pynlpl.formats.folia.Note method), 257 copychildren() (pynlpl.formats.folia.Observation method), 531 copychildren() copychildren() (pynlpl.formats.folia.CoreferenceLayer method), 566 method), 660 copychildren() (pynlpl.formats.folia.ObservationLayer (pynlpl.formats.folia.CoreferenceLink method), 696 copychildren() method), 766 copychildren() (pynlpl.formats.folia.Original method), copychildren() (pynlpl.formats.folia.Correction method), 988 945 copychildren() (pynlpl.formats.folia.Paragraph method), copychildren() (pynlpl.formats.folia.Current method), 270 955 copychildren() (pynlpl.formats.folia.Part method), 283 copychildren() (pynlpl.formats.folia.Definition method), (pynlpl.formats.folia.PhonContent copychildren() method), 507 116 (pynlpl.formats.folia.PosAnnotation copychildren() (pynlpl.formats.folia.DependenciesLayer copychildren() method), 672 method), 441 copychildren() (pynlpl.formats.folia.Dependency copychildren() (pynlpl.formats.folia.Predicate method), method), 543 578

copychildren() (pynlpl.formats.folia.Quote method), 296	CoreferenceChain (class in pynlpl.formats.folia), 527
copychildren() (pynlpl.formats.folia.Reference method),	CoreferenceLayer (class in pynlpl.formats.folia), 656
309	CoreferenceLink (class in pynlpl.formats.folia), 762
copychildren() (pynlpl.formats.folia.Row method), 322	Corpus (class in pynlpl.formats.sonar), 1054
copychildren() (pynlpl.formats.folia.SemanticRole	CorpusDocument (class in pynlpl.formats.sonar), 1054
method), 624	CorpusDocumentX (class in pynlpl.formats.sonar), 1054
copychildren() (pynlpl.formats.folia.SemanticRolesLayer	CorpusFiles (class in pynlpl.formats.sonar), 1055
method), 743	CorpusX (class in pynlpl.formats.sonar), 1055
copychildren() (pynlpl.formats.folia.SenseAnnotation method), 474	correct() (pynlpl.formats.folia.AbstractAnnotationLayer method), 77
copychildren() (pynlpl.formats.folia.Sentence method), 336	correct() (pynlpl.formats.folia.AbstractSpanAnnotation method), 54
copychildren() (pynlpl.formats.folia.Sentiment method), 589	correct() (pynlpl.formats.folia.AbstractStructureElement method), 40
copychildren() (pynlpl.formats.folia.SentimentLayer method), 707	correct() (pynlpl.formats.folia.AllowTokenAnnotation method), 49
copychildren() (pynlpl.formats.folia.Statement method),	correct() (pynlpl.formats.folia.Alternative method), 921
601	correct() (pynlpl.formats.folia.Cell method), 103
copychildren() (pynlpl.formats.folia.StatementLayer	correct() (pynlpl.formats.folia.Chunk method), 520
method), 719	correct() (pynlpl.formats.folia.ChunkingLayer method),
copychildren() (pynlpl.formats.folia.SubjectivityAnnotation	
method), 485	correct() (pynlpl.formats.folia.CoreferenceChain
copychildren() (pynlpl.formats.folia.Suggestion method),	method), 531
999	correct() (pynlpl.formats.folia.CoreferenceLayer
copychildren() (pynlpl.formats.folia.SynsetFeature	method), 660
method), 875	correct() (pynlpl.formats.folia.CoreferenceLink method),
copychildren() (pynlpl.formats.folia.SyntacticUnit	766
method), 613	correct() (pynlpl.formats.folia.Correction method), 945
copychildren() (pynlpl.formats.folia.SyntaxLayer	correct() (pynlpl.formats.folia.Current method), 955
method), 731	correct() (pynlpl.formats.folia.Definition method), 116
copychildren() (pynlpl.formats.folia.Table method), 350	correct() (pynlpl.formats.folia.DependenciesLayer
copychildren() (pynlpl.formats.folia.TableHead method),	method), 672
376	correct() (pynlpl.formats.folia.Dependency method), 543
copychildren() (pynlpl.formats.folia.Term method), 363	correct() (pynlpl.formats.folia.DependencyDependent
copychildren() (pynlpl.formats.folia.Text method), 389	method), 778
copychildren() (pynlpl.formats.folia.TextContent	correct() (pynlpl.formats.folia.Division method), 129
method), 497	correct() (pynlpl.formats.folia.EntitiesLayer method),
copychildren() (pynlpl.formats.folia.TextMarkupCorrection	correct() (pynlpl.formats.folia.Entity method), 555
method), 842 copychildren() (pynlpl.formats.folia.TextMarkupError	correct() (pynlpl.formats.folia.Entry method), 142
method), 852	correct() (pynlpl.formats.folia.Event method), 142
copychildren() (pynlpl.formats.folia.TextMarkupGap	correct() (pynlpl.formats.folia.Example method), 168
method), 810	correct() (pynlpl.formats.folia.Figure method), 181
copychildren() (pynlpl.formats.folia.TextMarkupString	correct() (pynlpl.formats.folia.Head method), 205
method), 820	correct() (pynlpl.formats.folia.Headspan method), 790
copychildren() (pynlpl.formats.folia.TextMarkupStyle	correct() (pynlpl.formats.folia.Linebreak method), 218
method), 831	correct() (pynlpl.formats.folia.List method), 231
copychildren() (pynlpl.formats.folia.TimeSegment	correct() (pynlpl.formats.folia.ListItem method), 244
method), 636	correct() (pynlpl.formats.folia.New method), 977
copychildren() (pynlpl.formats.folia.TimingLayer	correct() (pynlpl.formats.folia.Note method), 257
method), 754	correct() (pynlpl.formats.folia.Observation method), 566
copychildren() (pynlpl.formats.folia.Whitespace	correct() (pynlpl.formats.folia.ObservationLayer
method), 402	method), 696
copychildren() (pynlpl.formats.folia.Word method), 416	correct() (pynlpl.formats.folia.Paragraph method), 270

correct() (pynlpl.formats.folia.Part method), 283 correct() (pynlpl.formats.folia.Predicate method), 578 correct() (pynlpl.formats.folia.Quote method), 296	count() (pynlpl.formats.folia.Chunk method), 520 count() (pynlpl.formats.folia.ChunkingLayer method), 648
correct() (pynlpl.formats.folia.Reference method), 309 correct() (pynlpl.formats.folia.Row method), 322	count() (pynlpl.formats.folia.CoreferenceChain method), 531
correct() (pynlpl.formats.folia.SemanticRole method), 625	count() (pynlpl.formats.folia.CoreferenceLayer method), 660
correct() (pynlpl.formats.folia.SemanticRolesLayer method), 743	count() (pynlpl.formats.folia.CoreferenceLink method), 766
correct() (pynlpl.formats.folia.Sentence method), 336 correct() (pynlpl.formats.folia.Sentiment method), 590	count() (pynlpl.formats.folia.Correction method), 945 count() (pynlpl.formats.folia.Current method), 955
correct() (pynlpl.formats.folia.SentimentLayer method), 707	count() (pynlpl.formats.folia.Definition method), 116 count() (pynlpl.formats.folia.DependenciesLayer
correct() (pynlpl.formats.folia.Statement method), 601	method), 672
correct() (pynlpl.formats.folia.StatementLayer method),	count() (pynlpl.formats.folia.Dependency method), 543
719	count() (pynlpl.formats.folia.DependencyDependent
correct() (pynlpl.formats.folia.SyntacticUnit method),	method), 778
613	count() (pynlpl.formats.folia.Description method), 1033
correct() (pynlpl.formats.folia.SyntaxLayer method), 731	count() (pynlpl.formats.folia.Division method), 129
correct() (pynlpl.formats.folia.Table method), 350	count() (pynlpl.formats.folia.Document method), 17
correct() (pynlpl.formats.folia.TableHead method), 376 correct() (pynlpl.formats.folia.Term method), 363	count() (pynlpl.formats.folia.DomainAnnotation method), 430
correct() (pynlpl.formats.folia.Text method), 389	count() (pynlpl.formats.folia.EnddatetimeFeature
correct() (pynlpl.formats.folia.TimeSegment method),	method), 908
636	count() (pynlpl.formats.folia.EntitiesLayer method), 684
correct() (pynlpl.formats.folia.TimingLayer method), 754	count() (pynlpl.formats.folia.Entity method), 555
correct() (pynlpl.formats.folia.Whitespace method), 402	count() (pynlpl.formats.folia.Entry method), 142
correct() (pynlpl.formats.folia.Word method), 416	count() (pynlpl.formats.folia.ErrorDetection method),
Correction (class in pynlpl.formats.folia), 941	966
corrections() (pynlpl.formats.folia.Sentence method), 336	count() (pynlpl.formats.folia.Event method), 155
correctwords() (pynlpl.formats.folia.Sentence method),	count() (pynlpl.formats.folia.Example method), 168
336	count() (pynlpl.formats.folia.Feature method), 864
count() (pynlpl.formats.folia.AbstractAnnotationLayer method), 77	count() (pynlpl.formats.folia.Figure method), 181 count() (pynlpl.formats.folia.Gap method), 192
count() (pynlpl.formats.folia.AbstractElement method),	count() (pynlpl.formats.folia.Head method), 205
27	count() (pynlpl.formats.folia.Headspan method), 790
count() (pynlpl.formats.folia.AbstractSpanAnnotation	count() (pynlpl.formats.folia.LangAnnotation method),
method), 54	452
count() (pynlpl.formats.folia.AbstractStructureElement method), 40	$count()\ (pynlpl.formats.folia. Lemma Annotation\ method),$
	463
count() (pynlpl.formats.folia.AbstractTextMarkup	count() (pynlpl.formats.folia.Linebreak method), 218
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation method), 65	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244 count() (pynlpl.formats.folia.Metric method), 1044
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation method), 65 count() (pynlpl.formats.folia.ActorFeature method), 886	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244 count() (pynlpl.formats.folia.Metric method), 1044 count() (pynlpl.formats.folia.New method), 977 count() (pynlpl.formats.folia.Note method), 257 count() (pynlpl.formats.folia.Observation method), 566
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation method), 65 count() (pynlpl.formats.folia.ActorFeature method), 886 count() (pynlpl.formats.folia.Alignment method), 1010 count() (pynlpl.formats.folia.AlignReference method), 1022	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244 count() (pynlpl.formats.folia.Metric method), 1044 count() (pynlpl.formats.folia.New method), 977 count() (pynlpl.formats.folia.Note method), 257 count() (pynlpl.formats.folia.Observation method), 566 count() (pynlpl.formats.folia.ObservationLayer method),
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation method), 65 count() (pynlpl.formats.folia.ActorFeature method), 886 count() (pynlpl.formats.folia.Alignment method), 1010 count() (pynlpl.formats.folia.AlignReference method), 1022 count() (pynlpl.formats.folia.Alternative method), 922	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244 count() (pynlpl.formats.folia.Metric method), 1044 count() (pynlpl.formats.folia.New method), 977 count() (pynlpl.formats.folia.Note method), 257 count() (pynlpl.formats.folia.Observation method), 566 count() (pynlpl.formats.folia.ObservationLayer method), 696
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation method), 65 count() (pynlpl.formats.folia.ActorFeature method), 886 count() (pynlpl.formats.folia.Alignment method), 1010 count() (pynlpl.formats.folia.AlignReference method), 1022 count() (pynlpl.formats.folia.Alternative method), 922 count() (pynlpl.formats.folia.AlternativeLayers method),	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244 count() (pynlpl.formats.folia.Metric method), 1044 count() (pynlpl.formats.folia.New method), 977 count() (pynlpl.formats.folia.Note method), 257 count() (pynlpl.formats.folia.Observation method), 566 count() (pynlpl.formats.folia.ObservationLayer method), 696 count() (pynlpl.formats.folia.Original method), 988
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation method), 65 count() (pynlpl.formats.folia.ActorFeature method), 886 count() (pynlpl.formats.folia.Alignment method), 1010 count() (pynlpl.formats.folia.AlignReference method), 1022 count() (pynlpl.formats.folia.Alternative method), 922 count() (pynlpl.formats.folia.AlternativeLayers method), 933	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244 count() (pynlpl.formats.folia.Metric method), 1044 count() (pynlpl.formats.folia.New method), 977 count() (pynlpl.formats.folia.Note method), 257 count() (pynlpl.formats.folia.Observation method), 566 count() (pynlpl.formats.folia.ObservationLayer method), 696 count() (pynlpl.formats.folia.Original method), 988 count() (pynlpl.formats.folia.Paragraph method), 270
count() (pynlpl.formats.folia.AbstractTextMarkup method), 88 count() (pynlpl.formats.folia.AbstractTokenAnnotation method), 65 count() (pynlpl.formats.folia.ActorFeature method), 886 count() (pynlpl.formats.folia.Alignment method), 1010 count() (pynlpl.formats.folia.AlignReference method), 1022 count() (pynlpl.formats.folia.Alternative method), 922 count() (pynlpl.formats.folia.AlternativeLayers method),	count() (pynlpl.formats.folia.Linebreak method), 218 count() (pynlpl.formats.folia.List method), 231 count() (pynlpl.formats.folia.ListItem method), 244 count() (pynlpl.formats.folia.Metric method), 1044 count() (pynlpl.formats.folia.New method), 977 count() (pynlpl.formats.folia.Note method), 257 count() (pynlpl.formats.folia.Observation method), 566 count() (pynlpl.formats.folia.ObservationLayer method), 696 count() (pynlpl.formats.folia.Original method), 988

count() (pynlpl.formats.folia.PosAnnotation method), 441	deepvalidation() (pynlpl.formats.folia.AbstractAnnotationLayer method), 77
count() (pynlpl.formats.folia.Predicate method), 578 count() (pynlpl.formats.folia.Quote method), 296	deepvalidation() (pynlpl.formats.folia.AbstractElement method), 27
count() (pynlpl.formats.folia.Reference method), 309	deepvalidation() (pynlpl.formats.folia.AbstractSpanAnnotation
count() (pynlpl.formats.folia.Row method), 322	method), 54
count() (pynlpl.formats.folia.SemanticRole method), 625	$deep validation () \ (pynlpl.formats.folia. Abstract Structure Element$
count() (pynlpl.formats.folia.SemanticRolesLayer	method), 40
method), 743	deepvalidation() (pynlpl.formats.folia.AbstractTextMarkup
count() (pynlpl.formats.folia.SenseAnnotation method),	method), 88
474	deep validation ()  (pynlpl. for mats. folia. Abstract Token Annotation
count() (pynlpl.formats.folia.Sentence method), 336	method), 65
count() (pynlpl.formats.folia.Sentiment method), 590	deepvalidation() (pynlpl.formats.folia.ActorFeature
count() (pynlpl.formats.folia.SentimentLayer method),	method), 887
707	deepvalidation() (pynlpl.formats.folia.Alignment
count() (pynlpl.formats.folia.Statement method), 601	method), 1010
count() (pynlpl.formats.folia.StatementLayer method), 719	deepvalidation() (pynlpl.formats.folia.AlignReference method), 1022
count() (pynlpl.formats.folia.SubjectivityAnnotation method), 485	deepvalidation() (pynlpl.formats.folia.Alternative method), 922
count() (pynlpl.formats.folia.Suggestion method), 999	deepvalidation() (pynlpl.formats.folia.AlternativeLayers
count() (pynlpl.formats.folia.SynsetFeature method), 875	method), 933
count() (pynlpl.formats.folia.SyntacticUnit method), 613	deepvalidation() (pynlpl.formats.folia.BegindatetimeFeature
count() (pynlpl.formats.folia.SyntaxLayer method), 731	method), 898
count() (pynlpl.formats.folia.Table method), 350	deepvalidation() (pynlpl.formats.folia.Cell method), 103
count() (pynlpl.formats.folia.TableHead method), 376	deepvalidation() (pynlpl.formats.folia.Chunk method),
count() (pynlpl.formats.folia.Term method), 363	520
count() (pynlpl.formats.folia.Text method), 389	deepvalidation() (pynlpl.formats.folia.ChunkingLayer
count() (pynlpl.formats.folia.TextContent method), 497	method), 649
count() (pynlpl.formats.folia.TextMarkupCorrection	deepvalidation() (pynlpl.formats.folia.CoreferenceChain
method), 842	method), 531
count() (pynlpl.formats.folia.TextMarkupError method),	deepvalidation() (pynlpl.formats.folia.CoreferenceLayer
853	method), 660
count() (pynlpl.formats.folia.TextMarkupGap method),	deepvalidation() (pynlpl.formats.folia.CoreferenceLink
810	method), 766
count() (pynlpl.formats.folia.TextMarkupString method),	deepvalidation() (pynlpl.formats.folia.Correction
821	method), 946
count() (pynlpl.formats.folia.TextMarkupStyle method),	
831	955
count() (pynlpl.formats.folia.TimeSegment method), 636	deepvalidation() (pynlpl.formats.folia.Definition
count() (pynlpl.formats.folia.TimingLayer method), 754	method), 116
count() (pynlpl.formats.folia.Whitespace method), 402	deepvalidation() (pynlpl.formats.folia.DependenciesLayer
count() (pynlpl.formats.folia.Word method), 416	
	method), 672 deepvalidation() (pynlpl.formats.folia.Dependency
count() (pynlpl.statistics.FrequencyList method), 1063	1 " " " " " " " " " " " " " " " " " " "
create() (pynlpl.formats.folia.Document method), 17	method), 543
crude_tokenizer() (in module pynlpl.textprocessors), 1066	deepvalidation() (pynlpl.formats.folia.DependencyDependent method), 778
Current (class in pynlpl.formats.folia), 951	deepvalidation() (pynlpl.formats.folia.Description
current() (pynlpl.formats.folia.Correction method), 946	method), 1033
D	deepvalidation() (pynlpl.formats.folia.Division method),
D	129
date() (pynlpl.formats.folia.Document method), 17 declare() (pynlpl.formats.folia.Document method), 17	deepvalidation() (pynlpl.formats.folia.DomainAnnotation method), 430
declared() (pynlpl.formats.folia.Document method), 18	deepvalidation() (pynlpl.formats.folia.EnddatetimeFeature

- method), 909 deepvalidation() method), 684 deepvalidation() (pynlpl.formats.folia.Entity method), 555 deepvalidation() (pynlpl.formats.folia.Entry method), 142 deepvalidation() (pynlpl.formats.folia.ErrorDetection method), 966 deepvalidation() (pynlpl.formats.folia.Event method), 155 deepvalidation() (pynlpl.formats.folia.Example method), 168 deepvalidation() (pynlpl.formats.folia.Feature method), 864 deepvalidation() (pynlpl.formats.folia.Figure method), deepvalidation() (pynlpl.formats.folia.Gap method), 193 deepvalidation() (pynlpl.formats.folia.Head method), 205 deepvalidation() (pynlpl.formats.folia.Headspan method), 790 (pynlpl.formats.folia.LangAnnotation deepvalidation() method), 452 deepvalidation() (pynlpl.formats.folia.LemmaAnnotation method), 463 deepvalidation() (pynlpl.formats.folia.Linebreak method), 218 deepvalidation() (pynlpl.formats.folia.List method), 231 deepvalidation() (pynlpl.formats.folia.ListItem method), 244 deepvalidation() (pynlpl.formats.folia.Metric method), 1044 deepvalidation() (pynlpl.formats.folia.New method), 977 deepvalidation() (pynlpl.formats.folia.Note method), 257 (pynlpl.formats.folia.Observation deepvalidation() method), 566 deepvalidation() (pynlpl.formats.folia.ObservationLayer method), 696 deepvalidation() (pynlpl.formats.folia.Original method), 988 deepvalidation() (pynlpl.formats.folia.Paragraph method), 270 deepvalidation() (pynlpl.formats.folia.Part method), 283 deepvalidation() (pynlpl.formats.folia.PhonContent method), 507 deepvalidation() (pynlpl.formats.folia.PosAnnotation method), 441 deepvalidation() (pynlpl.formats.folia.Predicate method), 578 (pynlpl.formats.folia.Quote method), deepvalidation() 296 deepvalidation() (pynlpl.formats.folia.Reference method), 309 deepvalidation() (pynlpl.formats.folia.Row method), 322 deepvalidation() (pynlpl.formats.folia.SemanticRole defaultdatetime()
- method), 625 (pynlpl.formats.folia.EntitiesLayer deepvalidation() (pynlpl.formats.folia.SemanticRolesLayer method), 743 deepvalidation() (pynlpl.formats.folia.SenseAnnotation method), 474 deepvalidation() (pynlpl.formats.folia.Sentence method), 336 deepvalidation() (pynlpl.formats.folia.Sentiment method), 590 deepvalidation() (pynlpl.formats.folia.SentimentLayer method), 707 (pynlpl.formats.folia.Statement deepvalidation() method), 601 deepvalidation() (pynlpl.formats.folia.StatementLayer method), 719 deepvalidation() (pynlpl.formats.folia.SubjectivityAnnotation method), 485 (pynlpl.formats.folia.Suggestion deepvalidation() method), 999 deepvalidation() (pynlpl.formats.folia.SynsetFeature method), 875 deepvalidation() (pynlpl.formats.folia.SyntacticUnit method), 613 deepvalidation() (pynlpl.formats.folia.SyntaxLayer method), 731 deepvalidation() (pynlpl.formats.folia.Table method), 350 deepvalidation() (pynlpl.formats.folia.TableHead method), 376 deepvalidation() (pynlpl.formats.folia.Term method), 363 deepvalidation() (pynlpl.formats.folia.Text method), 389 (pynlpl.formats.folia.TextContent deepvalidation() method), 497 deepvalidation() (pynlpl.formats.folia.TextMarkupCorrection method), 842 deepvalidation() (pynlpl.formats.folia.TextMarkupError method), 853 deepvalidation() (pynlpl.formats.folia.TextMarkupGap method), 810 deepvalidation() (pynlpl.formats.folia.TextMarkupString method), 821 deepvalidation() (pynlpl.formats.folia.TextMarkupStyle method), 831 deepvalidation() (pynlpl.formats.folia.TimeSegment method), 636 deepvalidation() (pynlpl.formats.folia.TimingLayer method), 754 deepvalidation() (pynlpl.formats.folia.Whitespace method), 402 deepvalidation() (pynlpl.formats.folia.Word method), 416 defaultannotator() (pynlpl.formats.folia.Document method), 18 defaultannotatortype() (pynlpl.formats.folia.Document

method), 18

(pynlpl.formats.folia.Document

method), 19	946
$default parameters () \ (pynlpl. evaluation. Abstract Experiment$	description() (pynlpl.formats.folia.Current method), 955
method), 9	description ()  (pynlpl.formats.folia. Definition  method),
defaultset() (pynlpl.formats.folia.Document method), 19	116
Definition (class in pynlpl.formats.folia), 111	$description () \qquad (pynlpl.formats.folia. Dependencies Layer$
delete() (pynlpl.evaluation.AbstractExperiment method),	method), 672
9	description() (pynlpl.formats.folia.Dependency method),
deleteword() (pynlpl.formats.folia.Sentence method), 336	543
DependenciesLayer (class in pynlpl.formats.folia), 667 Dependency (class in pynlpl.formats.folia), 538	description() (pynlpl.formats.folia.DependencyDependent method), 778
Dependency (class in pyhipi.formats.folia), 338  DependencyDependent (class in pyhipl.formats.folia),	description() (pynlpl.formats.folia.Description method),
773	1033
dependent() (pynlpl.formats.folia.Dependency method),	description() (pynlpl.formats.folia.Division method), 129
543	description() (pynlpl.formats.folia.DomainAnnotation
depth() (pynlpl.datatypes.Trie method), 6	method), 430
depth() (pynlpl.search.AbstractSearchState method),	$description () \\ \hspace{0.5cm} (pynlpl.formats.folia. End date time Feature$
1059	method), 909
DepthFirstSearch (class in pynlpl.search), 1060	$description () \ (pynlpl. formats. folia. Entities Layer \ method),$
Description (class in pynlpl.formats.folia), 1029	684
description() (pynlpl.formats.folia.AbstractAnnotationLaye	
method), 77	description() (pynlpl.formats.folia.Entry method), 142 description() (pynlpl.formats.folia.ErrorDetection
description() (pynlpl.formats.folia.AbstractElement method), 27	description() (pynlpl.formats.folia.ErrorDetection method), 966
description() (pynlpl.formats.folia.AbstractSpanAnnotation	
method), 54	description() (pynlpl.formats.folia.Example method), 168
description() (pynlpl.formats.folia.AbstractStructureElemer	
method), 40	description() (pynlpl.formats.folia.Figure method), 181
description() (pynlpl.formats.folia.AbstractTextMarkup	description() (pynlpl.formats.folia.Gap method), 193
method), 88	description() (pynlpl.formats.folia.Head method), 205
description() (pynlpl.formats.folia.AbstractTokenAnnotatio	
method), 65	790
description() (pynlpl.formats.folia.ActorFeature method),	description() (pynlpl.formats.folia.LangAnnotation
description() (pynlpl.formats.folia.Alignment method),	method), 452 description() (pynlpl.formats.folia.LemmaAnnotation
1011	method), 463
description() (pynlpl.formats.folia.AlignReference	description() (pynlpl.formats.folia.Linebreak method),
method), 1022	218
description() (pynlpl.formats.folia.Alternative method),	description() (pynlpl.formats.folia.List method), 231
922	$description ()\ (pynlpl.formats.folia. ListItem\ method),\ 244$
description() (pynlpl.formats.folia.AlternativeLayers	description() (pynlpl.formats.folia.Metric method), 1044
method), 933	description() (pynlpl.formats.folia.New method), 977
description() (pynlpl.formats.folia.BegindatetimeFeature	description() (pynlpl.formats.folia.Note method), 257
method), 898	description() (pynlpl.formats.folia.Observation method),
description() (pynlpl.formats.folia.Cell method), 103 description() (pynlpl.formats.folia.Chunk method), 520	566 description() (pynlpl.formats.folia.ObservationLayer
description() (pynipiormats.folia.ChunkingLayer (pynipiormats.folia.ChunkingLayer	method), 696
method), 649	description() (pynlpl.formats.folia.Original method), 988
description() (pynlpl.formats.folia.CoreferenceChain	description() (pynlpl.formats.folia.Paragraph method),
method), 531	270
description() (pynlpl.formats.folia.CoreferenceLayer	description() (pynlpl.formats.folia.Part method), 283
method), 660	description() (pynlpl.formats.folia.PhonContent method),
description() (pynlpl.formats.folia.CoreferenceLink	507
method), 766	description() (pynlpl.formats.folia.PosAnnotation
description() (pynlpl.formats.folia.Correction method),	method), 441

description() (pynlpl.formats.folia.Predicate method), description() (pynlpl.formats.folia.Word method), 416 578 dict() (pynlpl.statistics.FrequencyList method), 1063 description() (pynlpl.formats.folia.Quote method), 296 Distribution (class in pynlpl.statistics), 1062 description() (pynlpl.formats.folia.Reference method), Division (class in pynlpl.formats.folia), 123 division() (pynlpl.formats.folia.Sentence method), 336 description() (pynlpl.formats.folia.Row method), 322 division() (pynlpl.formats.folia.Word method), 416 description() (pynlpl.formats.folia.SemanticRole Document (class in pynlpl.formats.folia), 14 method), 625 domain() (pynlpl.formats.folia.Word method), 416 description() (pynlpl.formats.folia.SemanticRolesLayer DomainAnnotation (class in pynlpl.formats.folia), 426 method), 743 done() (pynlpl.evaluation.AbstractExperiment method), 9 description() (pynlpl.formats.folia.SenseAnnotation dotproduct() (in module pynlpl.statistics), 1064 duration() (pynlpl.evaluation.AbstractExperiment method), 474 description() (pynlpl.formats.folia.Sentence method), 336 method), 9 description() (pynlpl.formats.folia.Sentiment method), Ε 590 description() (pynlpl.formats.folia.SentimentLayer EarlyEagerBeamSearch (class in pynlpl.search), 1060 method), 707 EnddatetimeFeature (class in pynlpl.formats.folia), 904 (pynlpl.formats.folia.Statement method), description() EntitiesLayer (class in pynlpl.formats.folia), 679 601 Entity (class in pynlpl.formats.folia), 550 (pynlpl.formats.folia.StatementLayer description() entropy() (pynlpl.statistics.Distribution method), 1062 method), 719 Entry (class in pynlpl.formats.folia), 136 description() (pynlpl.formats.folia.SubjectivityAnnotation Enum() (in module pynlpl.common), 3 method), 485 Error Detection (class in pynlpl.formats.folia), 962 description() (pynlpl.formats.folia.Suggestion method), Event (class in pynlpl.formats.folia), 149 999 Example (class in pynlpl.formats.folia), 162 description() (pynlpl.formats.folia.SynsetFeature expand() (pynlpl.search.AbstractSearchState method), method), 875 1060 (pynlpl.formats.folia.SyntacticUnit description() ExperimentPool (class in pynlpl.evaluation), 10 method), 613 extend() (pynlpl.datatypes.FIFOQueue method), 5 description() (pynlpl.formats.folia.SyntaxLayer method), extend() (pynlpl.datatypes.Queue method), 6 F description() (pynlpl.formats.folia.Table method), 350 description() (pynlpl.formats.folia.TableHead method), feat() (pynlpl.formats.folia.AbstractAnnotationLayer method), 77 description() (pynlpl.formats.folia.Term method), 363 feat() (pynlpl.formats.folia.AbstractElement method), 28 description() (pynlpl.formats.folia.Text method), 389 (pynlpl.formats.folia.AbstractSpanAnnotation feat() description() (pynlpl.formats.folia.TextContent method), method), 54 feat() (pynlpl.formats.folia.AbstractStructureElement description() (pynlpl.formats.folia.TextMarkupCorrection method), 40 method), 842 feat() (pynlpl.formats.folia.AbstractTextMarkup method), description() (pynlpl.formats.folia.TextMarkupError method), 853 feat() (pynlpl.formats.folia.AbstractTokenAnnotation (pynlpl.formats.folia.TextMarkupGap description() method), 65 method), 810 feat() (pynlpl.formats.folia.ActorFeature method), 887 (pynlpl.formats.folia.TextMarkupString description() feat() (pynlpl.formats.folia.Alignment method), 1011 method), 821 (pynlpl.formats.folia.AlignReference method), (pynlpl.formats.folia.TextMarkupStyle description() 1022 method), 831 feat() (pynlpl.formats.folia.Alternative method), 922 (pynlpl.formats.folia.TimeSegment description() feat() (pynlpl.formats.folia.AlternativeLayers method), method), 636 description() (pynlpl.formats.folia.TimingLayer method), feat() (pynlpl.formats.folia.BegindatetimeFeature method), 898 description() (pynlpl.formats.folia.Whitespace method), feat() (pynlpl.formats.folia.Cell method), 103 402 feat() (pynlpl.formats.folia.Chunk method), 520

```
feat() (pynlpl.formats.folia.ChunkingLayer method), 649
                                                           feat() (pynlpl.formats.folia.SemanticRolesLayer method),
feat() (pynlpl.formats.folia.CoreferenceChain method),
                                                                  (pynlpl.formats.folia.SenseAnnotation method),
                                                           feat()
feat() (pynlpl.formats.folia.CoreferenceLayer method),
                                                           feat() (pynlpl.formats.folia.Sentence method), 336
feat()
       (pynlpl.formats.folia.CoreferenceLink method),
                                                           feat() (pynlpl.formats.folia.Sentiment method), 590
                                                           feat() (pynlpl.formats.folia.SentimentLayer method), 708
feat() (pynlpl.formats.folia.Correction method), 946
                                                           feat() (pynlpl.formats.folia.Statement method), 601
feat() (pynlpl.formats.folia.Current method), 955
                                                           feat() (pynlpl.formats.folia.StatementLayer method), 719
feat() (pynlpl.formats.folia.Definition method), 116
                                                           feat()
                                                                       (pynlpl.formats.folia.SubjectivityAnnotation
feat() (pynlpl.formats.folia.DependenciesLayer method),
                                                                     method), 485
                                                           feat() (pynlpl.formats.folia.Suggestion method), 999
feat() (pynlpl.formats.folia.Dependency method), 543
                                                           feat() (pynlpl.formats.folia.SynsetFeature method), 875
            (pynlpl.formats.folia.DependencyDependent
                                                           feat() (pynlpl.formats.folia.SyntacticUnit method), 613
feat()
          method), 778
                                                           feat() (pynlpl.formats.folia.SyntaxLayer method), 731
feat() (pynlpl.formats.folia.Description method), 1033
                                                           feat() (pynlpl.formats.folia.Table method), 350
feat() (pynlpl.formats.folia.Division method), 129
                                                           feat() (pynlpl.formats.folia.TableHead method), 376
feat() (pynlpl.formats.folia.DomainAnnotation method),
                                                           feat() (pynlpl.formats.folia.Term method), 363
                                                           feat() (pynlpl.formats.folia.Text method), 389
feat() (pynlpl.formats.folia.EnddatetimeFeature method),
                                                           feat() (pynlpl.formats.folia.TextContent method), 497
                                                           feat()
                                                                        (pynlpl.formats.folia.TextMarkupCorrection
feat() (pynlpl.formats.folia.EntitiesLayer method), 684
feat() (pynlpl.formats.folia.Entity method), 555
                                                           feat() (pynlpl.formats.folia.TextMarkupError method),
feat() (pynlpl.formats.folia.Entry method), 142
feat() (pynlpl.formats.folia.ErrorDetection method), 966
                                                           feat() (pynlpl.formats.folia.TextMarkupGap method), 810
feat() (pynlpl.formats.folia.Event method), 155
                                                           feat() (pynlpl.formats.folia.TextMarkupString method),
feat() (pynlpl.formats.folia.Example method), 168
feat() (pynlpl.formats.folia.Feature method), 864
                                                           feat() (pynlpl.formats.folia.TextMarkupStyle method),
feat() (pynlpl.formats.folia.Figure method), 181
feat() (pynlpl.formats.folia.Gap method), 193
                                                           feat() (pynlpl.formats.folia.TimeSegment method), 636
                                                           feat() (pynlpl.formats.folia.TimingLayer method), 755
feat() (pynlpl.formats.folia.Head method), 205
feat() (pynlpl.formats.folia.Headspan method), 790
                                                           feat() (pynlpl.formats.folia.Whitespace method), 402
feat() (pynlpl.formats.folia.LangAnnotation method), 452
                                                           feat() (pynlpl.formats.folia.Word method), 416
feat() (pynlpl.formats.folia.LemmaAnnotation method),
                                                           Feature (class in pynlpl.formats.folia), 860
                                                           FIFOQueue (class in pynlpl.datatypes), 5
         463
feat() (pynlpl.formats.folia.Linebreak method), 218
                                                           Figure (class in pynlpl.formats.folia), 175
feat() (pynlpl.formats.folia.List method), 231
                                                           filesampler() (in module pynlpl.evaluation), 11
feat() (pynlpl.formats.folia.ListItem method), 244
                                                           find() (pynlpl.datatypes.Trie method), 6
feat() (pynlpl.formats.folia.Metric method), 1044
                                                           find keyword in context()
                                                                                                (in
                                                                                                            module
feat() (pynlpl.formats.folia.New method), 977
                                                                     pynlpl.textprocessors), 1066
feat() (pynlpl.formats.folia.Note method), 257
                                                           findcorrectionhandling() (pynlpl.formats.folia.AbstractAnnotationLayer
feat() (pynlpl.formats.folia.Observation method), 566
                                                                     method), 77
feat() (pynlpl.formats.folia.ObservationLayer method),
                                                           findcorrectionhandling() (pynlpl.formats.folia.AbstractElement
          696
                                                                     method), 28
feat() (pynlpl.formats.folia.Original method), 988
                                                           findcorrectionhandling() (pynlpl.formats.folia.AbstractSpanAnnotation
feat() (pynlpl.formats.folia.Paragraph method), 270
                                                                     method), 54
feat() (pynlpl.formats.folia.Part method), 283
                                                           findcorrectionhandling() (pynlpl.formats.folia.AbstractStructureElement
feat() (pynlpl.formats.folia.PhonContent method), 507
                                                                     method), 40
feat() (pynlpl.formats.folia.PosAnnotation method), 441
                                                           findcorrectionhandling() (pynlpl.formats.folia.AbstractTextMarkup
feat() (pynlpl.formats.folia.Predicate method), 578
                                                                     method), 88
feat() (pynlpl.formats.folia.Quote method), 296
                                                           find correction hand ling () \ (pynlpl. for mats. folia. Abstract Token Annotation
feat() (pynlpl.formats.folia.Reference method), 309
                                                                     method), 65
feat() (pynlpl.formats.folia.Row method), 322
                                                           findcorrectionhandling() (pynlpl.formats.folia.ActorFeature
feat() (pynlpl.formats.folia.SemanticRole method), 625
                                                                     method), 887
```

findcorrectionhandling() (pynlpl.formats.folia.Alignment findcorrectionhandling() (pynlpl.formats.folia.Feature method), 1011 (pynlpl.formats.folia.Feature
findcorrectionhandling() (pynlpl.formats.folia.AlignReferen@ndcorrectionhandling() (pynlpl.formats.folia.Figure method), 1022 method), 181
findcorrectionhandling() (pynlpl.formats.folia.Alternative findcorrectionhandling() (pynlpl.formats.folia.Gap method), 922 method), 193
findcorrectionhandling() (pynlpl.formats.folia.AlternativeLafiendcorrectionhandling() (pynlpl.formats.folia.Head method), 933 (pynlpl.formats.folia.Head method), 205
findcorrectionhandling() (pynlpl.formats.folia.Begindatetim <b>&amp;frekture</b> ectionhandling() (pynlpl.formats.folia.Headspan method), 898 method), 790
findcorrectionhandling() (pynlpl.formats.folia.Cell findcorrectionhandling() (pynlpl.formats.folia.LangAnnotation method), 103 method), 452
findcorrectionhandling() (pynlpl.formats.folia.Chunk findcorrectionhandling() (pynlpl.formats.folia.LemmaAnnotation method), 520 (pynlpl.formats.folia.Chunk findcorrectionhandling() (pynlpl.formats.folia.LemmaAnnotation method), 463
findcorrectionhandling() (pynlpl.formats.folia.ChunkingLay@mdcorrectionhandling() (pynlpl.formats.folia.Linebreak method), 649 method), 218
findcorrectionhandling() (pynlpl.formats.folia.CoreferenceChardcorrectionhandling() (pynlpl.formats.folia.List method), 532 method), 231
findcorrectionhandling() (pynlpl.formats.folia.CoreferenceLayelcorrectionhandling() (pynlpl.formats.folia.ListItem method), 661 method), 244
findcorrectionhandling() (pynlpl.formats.folia.CoreferenceLfindcorrectionhandling() (pynlpl.formats.folia.Metric method), 766 method), 1044
findcorrectionhandling() (pynlpl.formats.folia.Correction findcorrectionhandling() (pynlpl.formats.folia.New method), 946 method), 977
findcorrectionhandling() (pynlpl.formats.folia.Current findcorrectionhandling() (pynlpl.formats.folia.Note method), 955 (pynlpl.formats.folia.Note method), 257
findcorrectionhandling() (pynlpl.formats.folia.Definition method), 116 method), 567
findcorrectionhandling() (pynlpl.formats.folia.Dependencies <b>findco</b> rrectionhandling() (pynlpl.formats.folia.ObservationLayer method), 672 method), 696
findcorrectionhandling() (pynlpl.formats.folia.Dependency findcorrectionhandling() (pynlpl.formats.folia.Original method), 543 method), 988
findcorrectionhandling() (pynlpl.formats.folia.DependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependenc
findcorrectionhandling() (pynlpl.formats.folia.Description findcorrectionhandling() (pynlpl.formats.folia.Part method), 1033 (pynlpl.formats.folia.Part
findcorrectionhandling() (pynlpl.formats.folia.Division findcorrectionhandling() (pynlpl.formats.folia.PhonContent method), 129 method), 508
findcorrectionhandling() (pynlpl.formats.folia.DomainAnno fatiodrorrectionhandling() (pynlpl.formats.folia.PosAnnotation method), 430 method), 441
findcorrectionhandling() (pynlpl.formats.folia.EnddatetimeF <b>fantlue</b> rrectionhandling() (pynlpl.formats.folia.Predicate method), 909 method), 578
findcorrectionhandling() (pynlpl.formats.folia.EntitiesLayerfindcorrectionhandling() (pynlpl.formats.folia.Quote method), 684 (pynlpl.formats.folia.Quote method), 296
findcorrectionhandling() (pynlpl.formats.folia.Entity findcorrectionhandling() (pynlpl.formats.folia.Reference method), 555 method), 309
findcorrectionhandling() (pynlpl.formats.folia.Entry findcorrectionhandling() (pynlpl.formats.folia.Row method), 142 (pynlpl.formats.folia.Row method), 322
findcorrectionhandling() (pynlpl.formats.folia.ErrorDetectiofindcorrectionhandling() (pynlpl.formats.folia.SemanticRole method), 966 method), 625
findcorrectionhandling() (pynlpl.formats.folia.Event findcorrectionhandling() (pynlpl.formats.folia.SemanticRolesLayer method), 155 (pynlpl.formats.folia.Event findcorrectionhandling() (pynlpl.formats.folia.SemanticRolesLayer method), 743
findcorrectionhandling() (pynlpl.formats.folia.Example findcorrectionhandling() (pynlpl.formats.folia.SenseAnnotation method), 168 method), 474

- findcorrectionhandling() (pynlpl.formats.folia.Sentence findreplaceables() (pynlpl.formats.folia.AbstractElement method), 337 class method), 28 findcorrectionhandling() (pynlpl.formats.folia.Sentiment findreplaceables() (pynlpl.formats.folia.AbstractSpanAnnotation method), 590 class method), 54 findcorrectionhandling() (pynlpl.formats.folia.SentimentLayfandreplaceables() (pynlpl.formats.folia.AbstractStructureElement method), 708 class method), 40 findcorrectionhandling() (pynlpl.formats.folia.Statement findreplaceables() (pynlpl.formats.folia.AbstractTextMarkup method), 602 class method), 88 findcorrectionhandling() (pynlpl.formats.folia.StatementLayfindreplaceables() (pynlpl.formats.folia.AbstractTokenAnnotation class method), 65 method), 719 findcorrectionhandling() (pynlpl.formats.folia.SubjectivityAfindtatitaceables() (pynlpl.formats.folia.ActorFeature method), 485 class method), 887 findcorrectionhandling() (pynlpl.formats.folia.Suggestion findreplaceables() (pynlpl.formats.folia.Alignment class method), 999 method), 1011 findcorrectionhandling() (pynlpl.formats.folia.SynsetFeaturefindreplaceables() (pynlpl.formats.folia.AlignReference method), 876 class method), 1022 findcorrectionhandling() (pynlpl.formats.folia.SyntacticUnitfindreplaceables() (pynlpl.formats.folia.Alternative class method), 613 method), 922 findcorrectionhandling() (pynlpl.formats.folia.SyntaxLayer findreplaceables() (pynlpl.formats.folia.AlternativeLayers method), 731 class method), 933 findcorrectionhandling() (pynlpl.formats.folia.Table findreplaceables() (pynlpl.formats.folia.BegindatetimeFeature method), 350 class method), 898 findcorrectionhandling() (pynlpl.formats.folia.TableHead findreplaceables() (pynlpl.formats.folia.Cell class method), 376 method), 104 findcorrectionhandling() (pynlpl.formats.folia.Term findreplaceables() (pynlpl.formats.folia.Chunk class method), 363 method), 520 findcorrectionhandling() (pynlpl.formats.folia.Text findreplaceables() (pynlpl.formats.folia.ChunkingLayer method), 389 class method), 649 findcorrectionhandling() (pynlpl.formats.folia.TextContent findreplaceables() (pynlpl.formats.folia.CoreferenceChain method), 497 class method), 532 findcorrectionhandling() (pynlpl.formats.folia.TextMarkupC6nrdxtpbarceables() (pynlpl.formats.folia.CoreferenceLayer method), 842 class method), 661
  - findcorrectionhandling() (pynlpl.formats.folia.TextMarkupEfnodreplaceables() (pynlpl.formats.folia.CoreferenceLink method), 853 class method), 767
  - findcorrectionhandling() (pynlpl.formats.folia.TextMarkupGapdreplaceables() (pynlpl.formats.folia.Correction class method), 810 method), 946
  - findcorrectionhandling() (pynlpl.formats.folia.TextMarkupSfrintreplaceables() (pynlpl.formats.folia.Current class method), 821 method), 955
  - findcorrectionhandling() (pynlpl.formats.folia.TextMarkupSfyldreplaceables() (pynlpl.formats.folia.Definition class method), 832 method), 116
  - findcorrectionhandling() (pynlpl.formats.folia.TimeSegmentfindreplaceables() (pynlpl.formats.folia.DependenciesLayer method), 637 class method), 672
  - findcorrectionhandling() (pynlpl.formats.folia.TimingLayer findreplaceables() (pynlpl.formats.folia.Dependency method), 755 class method), 543
  - findcorrectionhandling() (pynlpl.formats.folia.Whitespace findreplaceables() (pynlpl.formats.folia.DependencyDependent method), 402 class method), 778
  - findcorrectionhandling() (pynlpl.formats.folia.Word findreplaceables() (pynlpl.formats.folia.Description class method), 416 (pynlpl.formats.folia.Word findreplaceables() (pynlpl.formats.folia.Description class method), 1033
  - finddefaultreference() (pynlpl.formats.folia.PhonContent findreplaceables() (pynlpl.formats.folia.Division class method), 508 method), 129
  - finddefaultreference() (pynlpl.formats.folia.TextContent findreplaceables() (pynlpl.formats.folia.DomainAnnotation method), 497 class method), 431
  - findreplaceables() (pynlpl.formats.folia.AbstractAnnotationIfancheplaceables() (pynlpl.formats.folia.EnddatetimeFeature class method), 77 class method), 909

findreplaceables() (pynlpl.formats.folia.EntitiesLayer findreplaceables() (pynlpl.formats.folia.Quote class class method), 684 method), 296 findreplaceables() (pynlpl.formats.folia.Entity class findreplaceables() (pynlpl.formats.folia.Reference class method), 555 method), 309 findreplaceables() (pynlpl.formats.folia.Entry class findreplaceables() (pynlpl.formats.folia.Row class method), 142 method), 322 findreplaceables() (pynlpl.formats.folia.ErrorDetection findreplaceables() (pynlpl.formats.folia.SemanticRole class method), 967 class method), 625 findreplaceables() (pynlpl.formats.folia.Event class findreplaceables() (pynlpl.formats.folia.SemanticRolesLayer class method), 743 method), 155 findreplaceables() (pynlpl.formats.folia.Example class findreplaceables() (pynlpl.formats.folia.SenseAnnotation class method), 475 method), 168 (pynlpl.formats.folia.Feature findreplaceables() (pynlpl.formats.folia.Sentence class findreplaceables() class method), 865 method), 337 findreplaceables() (pynlpl.formats.folia.Figure class findreplaceables() (pynlpl.formats.folia.Sentiment class method), 181 method), 590 findreplaceables() (pynlpl.formats.folia.Gap findreplaceables() (pynlpl.formats.folia.SentimentLayer class method), 193 class method), 708 findreplaceables() (pynlpl.formats.folia.Head findreplaceables() (pynlpl.formats.folia.Statement class class method), 205 method), 602 findreplaceables() (pynlpl.formats.folia.Headspan class findreplaceables() (pynlpl.formats.folia.StatementLayer method), 790 class method), 719 findreplaceables() (pynlpl.formats.folia.LangAnnotation findreplaceables() (pynlpl.formats.folia.SubjectivityAnnotation class method), 453 class method), 486 findreplaceables() (pynlpl.formats.folia.LemmaAnnotation findreplaceables() (pynlpl.formats.folia.Suggestion class class method), 464 method), 1000 findreplaceables() (pynlpl.formats.folia.Linebreak class findreplaceables() (pynlpl.formats.folia.SynsetFeature method), 218 class method), 876 findreplaceables() findreplaceables() (pynlpl.formats.folia.List class (pynlpl.formats.folia.SyntacticUnit method), 231 class method), 613 findreplaceables() (pynlpl.formats.folia.ListItem class findreplaceables() (pynlpl.formats.folia.SyntaxLayer method), 244 class method), 731 (pynlpl.formats.folia.Metric findreplaceables() class findreplaceables() (pynlpl.formats.folia.Table class method), 1045 method), 350 findreplaceables() (pynlpl.formats.folia.New class findreplaceables() (pynlpl.formats.folia.TableHead class method), 978 method), 376 findreplaceables() (pynlpl.formats.folia.Note class findreplaceables() (pynlpl.formats.folia.Term class method), 257 method), 363 (pynlpl.formats.folia.Text findreplaceables() (pynlpl.formats.folia.Observation class findreplaceables() class method), 567 method), 389 findreplaceables() (pynlpl.formats.folia.ObservationLayer findreplaceables() (pynlpl.formats.folia.TextContent class class method), 696 method), 497 findreplaceables() (pynlpl.formats.folia.Original findreplaceables() (pynlpl.formats.folia.TextMarkupCorrection class method), 989 class method), 842 findreplaceables() (pynlpl.formats.folia.Paragraph class findreplaceables() (pynlpl.formats.folia.TextMarkupError class method), 853 method), 270 (pynlpl.formats.folia.Part findreplaceables() (pynlpl.formats.folia.TextMarkupGap findreplaceables() class method), 283 class method), 810 findreplaceables() (pynlpl.formats.folia.TextMarkupString findreplaceables() (pynlpl.formats.folia.PhonContent class method), 821 class method), 508 findreplaceables() (pynlpl.formats.folia.PosAnnotation findreplaceables() (pynlpl.formats.folia.TextMarkupStyle class method), 442 class method), 832 findreplaceables() (pynlpl.formats.folia.Predicate class findreplaceables() (pynlpl.formats.folia.TimeSegment

Index 1109

class method), 637

method), 578

findreplaceables() (pynlpl.formats.folia.TimingLayer generate id() (pynlpl.formats.folia.CoreferenceLayer class method), 755 method), 661 findreplaceables() (pynlpl.formats.folia.Whitespace class generate id() (pynlpl.formats.folia.CoreferenceLink method), 402 method), 767 findreplaceables() (pynlpl.formats.folia.Word generate id() (pynlpl.formats.folia.Correction method), method), 416 findspan() (pynlpl.formats.folia.AbstractAnnotationLayer generate id() (pynlpl.formats.folia.Current method), 955 generate id() (pynlpl.formats.folia.Definition method), method), 77 findspan() (pynlpl.formats.folia.ChunkingLayer method), 117 649 generate\_id() (pynlpl.formats.folia.DependenciesLayer findspan() (pynlpl.formats.folia.CoreferenceLayer method), 673 generate\_id() (pynlpl.formats.folia.Dependency method), method), 661 (pynlpl.formats.folia.DependenciesLayer findspan() method), 673 generate\_id() (pynlpl.formats.folia.DependencyDependent findspan() (pynlpl.formats.folia.EntitiesLayer method), method), 778 684 generate\_id() (pynlpl.formats.folia.Division method), 129 findspan() (pynlpl.formats.folia.ObservationLayer (pynlpl.formats.folia.DomainAnnotation generate\_id() method), 696 method), 431 findspan() (pynlpl.formats.folia.SemanticRolesLayer generate id() (pynlpl.formats.folia.EntitiesLayer method), 743 method), 685 findspan() (pynlpl.formats.folia.SentimentLayer method), generate\_id() (pynlpl.formats.folia.Entity method), 555 generate\_id() (pynlpl.formats.folia.Entry method), 142 findspan() (pynlpl.formats.folia.StatementLayer method), (pynlpl.formats.folia.ErrorDetection generate\_id() method), 967 findspan() (pynlpl.formats.folia.SyntaxLayer method), generate\_id() (pynlpl.formats.folia.Event method), 155 generate id() (pynlpl.formats.folia.Example method), findspan() (pynlpl.formats.folia.TimingLayer method), 168 generate\_id() (pynlpl.formats.folia.Figure method), 182 findspans() (pynlpl.formats.folia.Word method), 417 generate\_id() (pynlpl.formats.folia.Head method), 206 findwords() (pynlpl.formats.folia.Document method), 19 generate\_id() (pynlpl.formats.folia.Headspan method), findwords() (pynlpl.formats.folia.Reader method), 805 790 fp\_rate() (pynlpl.evaluation.ClassEvaluation method), 9 generate\_id() (pynlpl.formats.folia.LangAnnotation FrequencyList (class in pynlpl.statistics), 1062 method), 453 fromstring() (pynlpl.datatypes.Pattern static method), 5 (pynlpl.formats.folia.LemmaAnnotation generate\_id() fscore() (pynlpl.evaluation.ClassEvaluation method), 9 method), 464 generate\_id() (pynlpl.formats.folia.Linebreak method), G 219 generate\_id() (pynlpl.formats.folia.List method), 231 Gap (class in pynlpl.formats.folia), 188  $generate\_id() \ (pynlpl.formats.folia. Abstract Annotation Laye generate\_id() \ (pynlpl.formats.folia. List I tem method), \ 244 tem (pynlpl.formats.folia) \ (pynlpl.for$ generate\_id() (pynlpl.formats.folia.New method), 978 method), 77  $generate\_id() \ (pynlpl.formats.folia. Abstract Span Annotation generate\_id() \ (pynlpl.formats.folia. Note \ method), 257$ generate id() (pynlpl.formats.folia.Observation method), method), 54 generate\_id() (pynlpl.formats.folia.AbstractStructureElement generate\_id() (pynlpl.formats.folia.ObservationLayer method), 40 method), 696 generate\_id() (pynlpl.formats.folia.AbstractTokenAnnotation generate\_id() (pynlpl.formats.folia.Original method), 989 method), 65 generate\_id() (pynlpl.formats.folia.Paragraph method), generate\_id() (pynlpl.formats.folia.Alternative method), 270 generate\_id() (pynlpl.formats.folia.Part method), 283 generate\_id() (pynlpl.formats.folia.Cell method), 104 (pynlpl.formats.folia.PosAnnotation generate\_id() generate id() (pynlpl.formats.folia.Chunk method), 520 method), 442 generate\_id() (pynlpl.formats.folia.ChunkingLayer generate\_id() (pynlpl.formats.folia.Predicate method), method), 649 578 (pynlpl.formats.folia.CoreferenceChain generate\_id() generate id() (pynlpl.formats.folia.Quote method), 296 method), 532

- generate\_id() (pynlpl.formats.folia.Reference method), 309 generate id() (pynlpl.formats.folia.Row method), 322 (pynlpl.formats.folia.SemanticRole generate\_id() getindex() method), 625 generate id() (pynlpl.formats.folia.SemanticRolesLayer method), 743 (pynlpl.formats.folia.SenseAnnotation generate\_id() method), 475 getindex() (pynlpl.formats.folia.Sentence method), generate\_id() 337 getindex() generate\_id() (pynlpl.formats.folia.Sentiment method), generate\_id() (pynlpl.formats.folia.SentimentLayer method), 708 generate\_id() (pynlpl.formats.folia.Statement method), 649 602 getindex() (pynlpl.formats.folia.StatementLayer generate id() method), 720 getindex() generate id() (pynlpl.formats.folia.SubjectivityAnnotation method), 486 getindex() generate\_id() (pynlpl.formats.folia.Suggestion method), 1000 generate\_id() (pynlpl.formats.folia.SyntacticUnit method), 613 generate\_id() (pynlpl.formats.folia.SyntaxLayer method), getindex() generate\_id() (pynlpl.formats.folia.Table method), 350 generate\_id() (pynlpl.formats.folia.TableHead method), 544 376 generate\_id() (pynlpl.formats.folia.Term method), 363 generate\_id() (pynlpl.formats.folia.Text method), 389 getindex() generate\_id() (pynlpl.formats.folia.TimeSegment 1034 method), 637 (pynlpl.formats.folia.TimingLayer generate\_id() getindex() method), 755 generate id() (pynlpl.formats.folia.Whitespace method), getindex() generate\_id() (pynlpl.formats.folia.Word method), 417 getalignedtarget() (pynlpl.formats.giza.GizaSentenceAlignment method), 1053 getcorrection() (pynlpl.formats.folia.Word method), 417 getcorrections() (pynlpl.formats.folia.Word method), 417 getindex() (pynlpl.formats.folia.AbstractAnnotationLayer 967 method), 77 (pynlpl.formats.folia.AbstractElement getindex() method), 28 getindex() (pynlpl.formats.folia.AbstractSpanAnnotation method), 54 getindex() (pynlpl.formats.folia.AbstractStructureElement method), 40 getindex() (pynlpl.formats.folia.AbstractTextMarkup getindex() method), 88 method), 453 getindex() (pynlpl.formats.folia.AbstractTokenAnnotation getindex() (pynlpl.formats.folia.LemmaAnnotation
- method), 65 getindex() (pynlpl.formats.folia.ActorFeature method), (pynlpl.formats.folia.Alignment method), getindex() (pynlpl.formats.folia.AlignReference method), getindex() (pynlpl.formats.folia.Alternative method), 922 (pynlpl.formats.folia.AlternativeLayers method), 933 (pynlpl.formats.folia.BegindatetimeFeature method), 898 getindex() (pynlpl.formats.folia.Cell method), 104 getindex() (pynlpl.formats.folia.Chunk method), 520 getindex() (pynlpl.formats.folia.ChunkingLayer method), (pynlpl.formats.folia.CoreferenceChain method), 532 (pynlpl.formats.folia.CoreferenceLayer method), 661 (pynlpl.formats.folia.CoreferenceLink method), 767 getindex() (pynlpl.formats.folia.Correction method), 946 getindex() (pynlpl.formats.folia.Current method), 956 getindex() (pynlpl.formats.folia.Definition method), 117 (pynlpl.formats.folia.DependenciesLayer method), 673 getindex() (pynlpl.formats.folia.Dependency method), getindex() (pynlpl.formats.folia.DependencyDependent method), 778 (pynlpl.formats.folia.Description method), getindex() (pynlpl.formats.folia.Division method), 130 (pynlpl.formats.folia.DomainAnnotation method), 431 (pynlpl.formats.folia.EnddatetimeFeature method), 909 getindex() (pynlpl.formats.folia.EntitiesLayer method), getindex() (pynlpl.formats.folia.Entity method), 555 getindex() (pynlpl.formats.folia.Entry method), 142 getindex() (pynlpl.formats.folia.ErrorDetection method), getindex() (pynlpl.formats.folia.Event method), 156 getindex() (pynlpl.formats.folia.Example method), 168 getindex() (pynlpl.formats.folia.Feature method), 865 getindex() (pynlpl.formats.folia.Figure method), 182 getindex() (pynlpl.formats.folia.Gap method), 193 getindex() (pynlpl.formats.folia.Head method), 206 getindex() (pynlpl.formats.folia.Headspan method), 790 (pynlpl.formats.folia.LangAnnotation

method), 464	method), 853
getindex() (pynlpl.formats.folia.Linebreak method), 219	getindex() (pynlpl.formats.folia.TextMarkupGap
getindex() (pynlpl.formats.folia.List method), 231	method), 810
getindex() (pynlpl.formats.folia.ListItem method), 244	getindex() (pynlpl.formats.folia.TextMarkupString
getindex() (pynlpl.formats.folia.Metric method), 1045	method), 821
getindex() (pynlpl.formats.folia.New method), 978	getindex() (pynlpl.formats.folia.TextMarkupStyle
getindex() (pynlpl.formats.folia.Note method), 257	method), 832
getindex() (pynlpl.formats.folia.Observation method),	getindex() (pynlpl.formats.folia.TimeSegment method),
567	637
getindex() (pynlpl.formats.folia.ObservationLayer method), 696	getindex() (pynlpl.formats.folia.TimingLayer method), 755
getindex() (pynlpl.formats.folia.Original method), 989	getindex() (pynlpl.formats.folia.Whitespace method), 402
getindex() (pynlpl.formats.folia.Paragraph method), 271	getindex() (pynlpl.formats.folia.Word method), 417
getindex() (pynlpl.formats.folia.Part method), 284	getmetadata() (pynlpl.formats.folia.AbstractAnnotationLayer
getindex() (pynlpl.formats.folia.PhonContent method),	method), 77
508	getmetadata() (pynlpl.formats.folia.AbstractElement
getindex() (pynlpl.formats.folia.PosAnnotation method),	method), 28
442 getindex() (pynlpl.formats.folia.Predicate method), 578	getmetadata() (pynlpl.formats.folia.AbstractSpanAnnotation method), 54
getindex() (pynlpl.formats.folia.Predicate method), 378 getindex() (pynlpl.formats.folia.Quote method), 296	getmetadata() (pynlpl.formats.folia.AbstractStructureElement
· · · · · · · · · · · · · · · · · · ·	method), 41
getindex() (pynlpl.formats.folia.Reference method), 310	
getindex() (pynlpl.formats.folia.Row method), 322	getmetadata() (pynlpl.formats.folia.AbstractTextMarkup
getindex() (pynlpl.formats.folia.SemanticRole method),	method), 88
625	getmetadata() (pynlpl.formats.folia.AbstractTokenAnnotation
getindex() (pynlpl.formats.folia.SemanticRolesLayer	method), 65
method), 743	getmetadata() (pynlpl.formats.folia.ActorFeature
getindex() (pynlpl.formats.folia.SenseAnnotation	method), 887
method), 475	getmetadata() (pynlpl.formats.folia.Alignment method),
getindex() (pynlpl.formats.folia.Sentence method), 337	1011
getindex() (pynlpl.formats.folia.Sentiment method), 590	getmetadata() (pynlpl.formats.folia.AlignReference
getindex() (pynlpl.formats.folia.SentimentLayer method),	method), 1022
708	getmetadata() (pynlpl.formats.folia.Alternative method),
getindex() (pynlpl.formats.folia.Statement method), 602	922
getindex() (pynlpl.formats.folia.StatementLayer method), 720	getmetadata() (pynlpl.formats.folia.AlternativeLayers method), 933
getindex() (pynlpl.formats.folia.SubjectivityAnnotation	getmetadata() (pynlpl.formats.folia.BegindatetimeFeature
method), 486	method), 898
getindex() (pynlpl.formats.folia.Suggestion method),	
1000	getmetadata() (pynlpl.formats.folia.Chunk method), 520
<pre>getindex() (pynlpl.formats.folia.SynsetFeature method),</pre>	getmetadata() (pynlpl.formats.folia.ChunkingLayer
876	method), 649
getindex() (pynlpl.formats.folia.SyntacticUnit method),	getmetadata() (pynlpl.formats.folia.CoreferenceChain
613	method), 532
getindex() (pynlpl.formats.folia.SyntaxLayer method),	getmetadata() (pynlpl.formats.folia.CoreferenceLayer
732	method), 661
getindex() (pynlpl.formats.folia.Table method), 350	getmetadata() (pynlpl.formats.folia.CoreferenceLink
getindex() (pynlpl.formats.folia.TableHead method), 376	method), 767
getindex() (pynlpl.formats.folia.Term method), 363	getmetadata() (pynlpl.formats.folia.Correction method),
getindex() (pynlpl.formats.folia.Text method), 389	946
getindex() (pynlpl.formats.folia.Text inclind), 369 getindex() (pynlpl.formats.folia.TextContent method),	getmetadata() (pynlpl.formats.folia.Current method), 956
497	getmetadata() (pynlpl.formats.folia.Definition method),
getindex() (pynlpl.formats.folia.TextMarkupCorrection	117
method), 842	getmetadata() (pynlpl.formats.folia.DependenciesLayer
getindex() (pynlpl.formats.folia.TextMarkupError	method), 673
Section ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) (	1110011001, 010

- getmetadata() (pynlpl.formats.folia.Dependency method), getmetadata() (pynlpl.formats.folia.DependencyDependent method), 778 getmetadata() (pynlpl.formats.folia.Description method), getmetadata() (pynlpl.formats.folia.Division method), 130 getmetadata() (pynlpl.formats.folia.DomainAnnotation method), 431 getmetadata() (pynlpl.formats.folia.EnddatetimeFeature method), 909 (pynlpl.formats.folia.EntitiesLayer getmetadata() method), 685 getmetadata() (pynlpl.formats.folia.Entity method), 555 getmetadata() (pynlpl.formats.folia.Entry method), 143 getmetadata() (pynlpl.formats.folia.ErrorDetection method), 967 getmetadata() (pynlpl.formats.folia.Event method), 156 getmetadata() (pynlpl.formats.folia.Example method), getmetadata() (pynlpl.formats.folia.Feature method), 865 getmetadata() (pynlpl.formats.folia.Figure method), 182 getmetadata() (pynlpl.formats.folia.Gap method), 193 getmetadata() (pynlpl.formats.folia.Head method), 206 getmetadata() (pynlpl.formats.folia.Headspan method), getmetadata() (pynlpl.formats.folia.LangAnnotation method), 453 (pynlpl.formats.folia.LemmaAnnotation getmetadata() method), 464 getmetadata() (pynlpl.formats.folia.Linebreak method), 219 getmetadata() (pynlpl.formats.folia.List method), 232 (pynlpl.formats.folia.ListItem method), getmetadata() getmetadata() (pynlpl.formats.folia.Metric method), 1045 getmetadata() (pynlpl.formats.folia.New method), 978 getmetadata() (pynlpl.formats.folia.Note method), 258 getmetadata() (pynlpl.formats.folia.Observation method), getmetadata() (pynlpl.formats.folia.ObservationLayer method), 696 (pynlpl.formats.folia.Original method), getmetadata() 989 getmetadata() (pynlpl.formats.folia.Paragraph method), 271 getmetadata() (pynlpl.formats.folia.Part method), 284 (pynlpl.formats.folia.PhonContent getmetadata() method), 508 (pynlpl.formats.folia.PosAnnotation getmetadata() method), 442 getmetadata() (pynlpl.formats.folia.Predicate method), 578
- getmetadata() (pynlpl.formats.folia.Quote method), 297 getmetadata() (pynlpl.formats.folia.Reference method), getmetadata() (pynlpl.formats.folia.Row method), 323 getmetadata() (pynlpl.formats.folia.SemanticRole method), 625 getmetadata() (pynlpl.formats.folia.SemanticRolesLayer method), 743 getmetadata() (pynlpl.formats.folia.SenseAnnotation method), 475 getmetadata() (pynlpl.formats.folia.Sentence method), 337 getmetadata() (pynlpl.formats.folia.Sentiment method), 590 getmetadata() (pynlpl.formats.folia.SentimentLayer method), 708 getmetadata() (pynlpl.formats.folia.Statement method), 602 getmetadata() (pynlpl.formats.folia.StatementLayer method), 720 getmetadata() (pynlpl.formats.folia.SubjectivityAnnotation method), 486 getmetadata() (pynlpl.formats.folia.Suggestion method), 1000 (pynlpl.formats.folia.SynsetFeature getmetadata() method), 876 getmetadata() (pynlpl.formats.folia.SyntacticUnit method), 613 (pynlpl.formats.folia.SyntaxLayer getmetadata() method), 732 getmetadata() (pynlpl.formats.folia.Table method), 350 getmetadata() (pynlpl.formats.folia.TableHead method), getmetadata() (pynlpl.formats.folia.Term method), 363 getmetadata() (pynlpl.formats.folia.Text method), 389 getmetadata() (pynlpl.formats.folia.TextContent method), getmetadata() (pynlpl.formats.folia.TextMarkupCorrection method), 842 getmetadata() (pynlpl.formats.folia.TextMarkupError method), 853 getmetadata() (pynlpl.formats.folia.TextMarkupGap method), 810 (pynlpl.formats.folia.TextMarkupString getmetadata() method), 821 (pynlpl.formats.folia.TextMarkupStyle getmetadata() method), 832 (pynlpl.formats.folia.TimeSegment getmetadata() method), 637 (pynlpl.formats.folia.TimingLayer getmetadata() method), 755 getmetadata() (pynlpl.formats.folia.Whitespace method),

getmetadata() (pynlpl.formats.folia.Word method), 417

getreference()	(pynlpl.formats.folia.PhonContent	130		
method), 50	18	gettextdelimiter()	(pynlpl.formats.folia.DomainAnnotation	n
getreference()	(pynlpl.formats.folia.TextContent	method)	, 431	
method), 49	7	gettextdelimiter()	(pynlpl.formats.folia.EnddatetimeFeatu	ıre
gettextdelimiter() (pyr	nlpl.formats.folia.AbstractAnnotation	Layer method)	, 909	
method), 77	·	gettextdelimiter()	(pynlpl.formats.folia.EntitiesLaye	r
gettextdelimiter() (p	ynlpl.formats.folia.AbstractElement	method)		
method), 28		,	(pynlpl.formats.folia.Entity method)	
* * * * * * * * * * * * * * * * * * * *	nlpl.formats.folia.AbstractSpanAnnot	-	(1) 1	,
method), 54		gettextdelimiter()	(pynlpl.formats.folia.Entry method)	
	nlpl.formats.folia.AbstractStructureEl	-	45 1	,
method), 41		gettextdelimiter()	(pynlpl.formats.folia.ErrorDetection	า
	nlpl.formats.folia.AbstractTextMarku	~		-
method), 88		gettextdelimiter()		,
	nlpl.formats.folia.AbstractTokenAnno		(pyimpinorimatsironai.2 vent interiou)	,
method), 66	=	gettextdelimiter()	(pynlpl.formats.folia.Example	<b>a</b>
gettextdelimiter()	(pynlpl.formats.folia.ActorFeature	method)	** *	_
method), 88			(pynlpl.formats.folia.Feature method)	
gettextdelimiter()	(pynlpl.formats.folia.Alignment	865	(pympi.ioimats.iona.i eature method)	,
method), 10	= - =		(numbel formats folio Figure mathed)	
* * * * * * * * * * * * * * * * * * * *		182	(pynlpl.formats.folia.Figure method)	,
	bynlpl.formats.folia.AlignReference		(1-1 f	2
method), 10		-	(pynlpl.formats.folia.Gap method), 193	
gettextdelimiter()	(pynlpl.formats.folia.Alternative	gettextdelimiter()	(pynlpl.formats.folia.Head method)	,
method), 92		206		
	nlpl.formats.folia.AlternativeLayers	gettextdelimiter()	(pynlpl.formats.folia.Headspar	1
method), 93		method)		
	-	_	(pynlpl.formats.folia.LangAnnotation	1
method), 89		method)		
	nlpl.formats.folia.Cell method), 104	_	(pynlpl.formats.folia.LemmaAnnotatio	n
	vnlpl.formats.folia.Chunk method),	method)		
520		gettextdelimiter()	(pynlpl.formats.folia.Linebreal	ς.
	pynlpl.formats.folia.ChunkingLayer	method)		
method), 64		-	(pynlpl.formats.folia.List method), 232	
	nlpl.formats.folia.CoreferenceChain	gettextdelimiter()	(pynlpl.formats.folia.ListItem method)	,
method), 53		245		
	nlpl.formats.folia.CoreferenceLayer	gettextdelimiter()	(pynlpl.formats.folia.Metric method)	,
method), 66		1045		
gettextdelimiter() (py	ynlpl.formats.folia.CoreferenceLink	<pre>gettextdelimiter()</pre>	(pynlpl.formats.folia.New method)	,
method), 76		978		
gettextdelimiter()	(pynlpl.formats.folia.Correction	gettextdelimiter()	(pynlpl.formats.folia.Note method)	,
method), 94	.6	258		
gettextdelimiter() (py	nlpl.formats.folia.Current method),	<pre>gettextdelimiter()</pre>	(pynlpl.formats.folia.Observation	1
956		method)		
gettextdelimiter()	(pynlpl.formats.folia.Definition		(pynlpl.formats.folia.ObservationLaye	r
method), 11	** *	method)		
			(pynlpl.formats.folia.Original method)	
method), 67	= = = = = = = = = = = = = = = = = = = =	989	(1) I	,
gettextdelimiter()	(pynlpl.formats.folia.Dependency		(pynlpl.formats.folia.Paragrapl	า
method), 54	** *	method)		•
* * * * * * * * * * * * * * * * * * * *			(pynlpl.formats.folia.Part method), 284	1
method), 77		gettextdelimiter()	(pynlpl.formats.folia.PhonConten	
gettextdelimiter()	(pynlpl.formats.folia.Description	method)		٠
method), 10		gettextdelimiter()	(pynlpl.formats.folia.PosAnnotation	า
	nlpl.formats.folia.Division method),	method)	= - =	4
5-monutation () (Pyl		memou)	, · · <del>-</del>	

gettextdelimiter() (pynlpl.formats.folia.Predicate method), 579	method), 637 gettextdelimiter() (pynlpl.formats.folia.TimingLayer
gettextdelimiter() (pynlpl.formats.folia.Quote method), 297	method), 755 gettextdelimiter() (pynlpl.formats.folia.Whitespace
gettextdelimiter() (pynlpl.formats.folia.Reference method), 310	method), 402 gettextdelimiter() (pynlpl.formats.folia.Word method),
gettextdelimiter() (pynlpl.formats.folia.Row method), 323	417 GizaModel (class in pynlpl.formats.giza), 1053
gettextdelimiter() (pynlpl.formats.folia.SemanticRole	GizaSentenceAlignment (class in pynlpl.formats.giza),
gettextdelimiter() (pynlpl.formats.folia.SemanticRolesLaye method), 743	er H
gettextdelimiter() (pynlpl.formats.folia.SenseAnnotation method), 475	hasannotation() (pynlpl.formats.folia.AbstractAnnotationLayer method), 77
gettextdelimiter() (pynlpl.formats.folia.Sentence method), 337	hasannotation() (pynlpl.formats.folia.AbstractSpanAnnotation method), 54
gettextdelimiter() (pynlpl.formats.folia.Sentiment method), 590	hasannotation() (pynlpl.formats.folia.AbstractStructureElement method), 41
gettextdelimiter() (pynlpl.formats.folia.SentimentLayer method), 708	hasannotation() (pynlpl.formats.folia.AllowTokenAnnotation method), 49
gettextdelimiter() (pynlpl.formats.folia.Statement method), 602	hasannotation() (pynlpl.formats.folia.Alternative method), 922
gettextdelimiter() (pynlpl.formats.folia.StatementLayer method), 720	hasannotation() (pynlpl.formats.folia.Cell method), 104 hasannotation() (pynlpl.formats.folia.Chunk method),
gettextdelimiter() (pynlpl.formats.folia.SubjectivityAnnota method), 486	
gettextdelimiter() (pynlpl.formats.folia.Suggestion method), 1000	method), 649 hasannotation() (pynlpl.formats.folia.CoreferenceChain
gettextdelimiter() (pynlpl.formats.folia.SynsetFeature method), 876	method), 532 hasannotation() (pynlpl.formats.folia.CoreferenceLayer
gettextdelimiter() (pynlpl.formats.folia.SyntacticUnit method), 614	method), 661 hasannotation() (pynlpl.formats.folia.CoreferenceLink
gettextdelimiter() (pynlpl.formats.folia.SyntaxLayer method), 732	method), 767 hasannotation() (pynlpl.formats.folia.Definition method),
gettextdelimiter() (pynlpl.formats.folia.Table method), 350	117 hasannotation() (pynlpl.formats.folia.DependenciesLayer
gettextdelimiter() (pynlpl.formats.folia.TableHead method), 376	method), 673 hasannotation() (pynlpl.formats.folia.Dependency
gettextdelimiter() (pynlpl.formats.folia.Term method), 363	method), 544 hasannotation() (pynlpl.formats.folia.DependencyDependent
gettextdelimiter() (pynlpl.formats.folia.Text method), 389 gettextdelimiter() (pynlpl.formats.folia.TextContent	method), 779 hasannotation() (pynlpl.formats.folia.Division method),
method), 497 gettextdelimiter() (pynlpl.formats.folia.TextMarkupCorrect	130
method), 843 gettextdelimiter() (pynlpl.formats.folia.TextMarkupError	method), 685 hasannotation() (pynlpl.formats.folia.Entity method), 555
method), 853 gettextdelimiter() (pynlpl.formats.folia.TextMarkupGap	hasannotation() (pynlpl.formats.folia.Entry method), 143
method), 811	hasannotation() (pynlpl.formats.folia.Event method), 156 hasannotation() (pynlpl.formats.folia.Example method),
gettextdelimiter() (pynlpl.formats.folia.TextMarkupString method), 821	hasannotation() (pynlpl.formats.folia.Figure method),
gettextdelimiter() (pynlpl.formats.folia.TextMarkupStyle method), 832	182 hasannotation() (pynlpl.formats.folia.Head method), 206
gettextdelimiter() (pynlpl.formats.folia.TimeSegment	

hasannotation() (pynlpl.formats.folia.Headspan method),	104	( 116 (61 D 6 W
790 hasannatation() (numbra formats folia Linabraak mathod)	hasannotationlayer()	(pynlpl.formats.folia.Definition
hasannotation() (pynlpl.formats.folia.Linebreak method), 219	method), 117 hasannotationlayer()	(pynlpl.formats.folia.Division
hasannotation() (pynlpl.formats.folia.List method), 232	method), 130	(pyinpi:formats.fona.Division
hasannotation() (pynlpl.formats.folia.ListItem method),		nlpl.formats.folia.Entry method),
245	143	
hasannotation() (pynlpl.formats.folia.Note method), 258	hasannotationlayer()	(pynlpl.formats.folia.Event
hasannotation() (pynlpl.formats.folia.Observation	method), 156	
method), 567	hasannotationlayer()	(pynlpl.formats.folia.Example
hasannotation() (pynlpl.formats.folia.ObservationLayer	method), 169	( 1 - 1 C 4 - C - 1' - E'
method), 696 hasannetation() (nyalpl formats folio Porograph method)	hasannotationlayer()	(pynlpl.formats.folia.Figure
hasannotation() (pynlpl.formats.folia.Paragraph method),	method), 182	nlpl.formats.folia.Head method),
hasannotation() (pynlpl.formats.folia.Part method), 284	206	inpi.iormats.iona.riead method),
hasannotation() (pynlpl.formats.folia.Predicate method),	hasannotationlayer()	(pynlpl.formats.folia.Linebreak
579	method), 219	(p) inpiriorina is irona. Emecrean
hasannotation() (pynlpl.formats.folia.Quote method), 297		ynlpl.formats.folia.List method),
hasannotation() (pynlpl.formats.folia.Reference method),	232	, ,
310	hasannotationlayer()	(pynlpl.formats.folia.ListItem
hasannotation() (pynlpl.formats.folia.Row method), 323	method), 245	
hasannotation() (pynlpl.formats.folia.SemanticRole	hasannotationlayer() (py	nlpl.formats.folia.Note method),
method), 625	258	
hasannotation() (pynlpl.formats.folia.SemanticRolesLayer		(pynlpl.formats.folia.Paragraph
method), 743	method), 271	
hasannotation() (pynlpl.formats.folia.Sentence method), 337	hasannotationlayer() (py	ynlpl.formats.folia.Part method),
hasannotation() (pynlpl.formats.folia.Sentiment method),	hasannotationlayer()	(pynlpl.formats.folia.Quote
590	method), 297	
$has annotation () \\ \hspace{0.5cm} (pynlpl.formats.folia. Sentiment Layer$	hasannotationlayer()	(pynlpl.formats.folia.Reference
method), 708	method), 310	
hasannotation() (pynlpl.formats.folia.Statement method),		vnlpl.formats.folia.Row method),
602	323	( 116 61 6
hasannotation() (pynlpl.formats.folia.StatementLayer	hasannotationlayer()	(pynlpl.formats.folia.Sentence
method), 720 hasannotation() (pynlpl.formats.folia.SyntacticUnit	method), 337	nlpl.formats.folia.Table method),
method), 614	351	inpr.formats.fona.fable method),
hasannotation() (pynlpl.formats.folia.SyntaxLayer	hasannotationlayer()	(pynlpl.formats.folia.TableHead
method), 732	method), 377	
hasannotation() (pynlpl.formats.folia.Table method), 350	hasannotationlayer() (py	nlpl.formats.folia.Term method),
hasannotation() (pynlpl.formats.folia.TableHead	364	
method), 376	hasannotationlayer() (py	nlpl.formats.folia.Text method),
hasannotation() (pynlpl.formats.folia.Term method), 363	390	
hasannotation() (pynlpl.formats.folia.Text method), 390		(pynlpl.formats.folia.Whitespace
hasannotation() (pynlpl.formats.folia.TimeSegment	method), 403	
method), 637	•	nlpl.formats.folia.Word method),
hasannotation() (pynlpl.formats.folia.TimingLayer	417	masta falia Commation mathed)
method), 755 hasannotation() (pynlpl.formats.folia.Whitespace	946	rmats.folia.Correction method),
method), 403		s.folia.Correction method), 946
hasannotation() (pynlpl.formats.folia.Word method), 417		rmats.folia.Correction method),
hasannotationlayer() (pynlpl.formats.folia.AbstractStructur		mediod),
method), 41		ts.folia.AbstractAnnotationLayer
hasannotationlayer() (pynlpl.formats.folia.Cell method),	method), 78	, and the second

- $\begin{array}{c} has phon() & (pynlpl.formats.folia. Abstract Element\\ method), 28 \end{array}$
- hasphon() (pynlpl.formats.folia.AbstractSpanAnnotation method), 55
- hasphon() (pynlpl.formats.folia.AbstractStructureElement method), 41
- hasphon() (pynlpl.formats.folia.AbstractTextMarkup method), 88
- hasphon() (pynlpl.formats.folia.AbstractTokenAnnotation method), 66
- hasphon() (pynlpl.formats.folia.ActorFeature method), 887
- hasphon() (pynlpl.formats.folia.Alignment method), 1011 hasphon() (pynlpl.formats.folia.AlignReference method), 1022
- hasphon() (pynlpl.formats.folia.Alternative method), 922 hasphon() (pynlpl.formats.folia.AlternativeLayers method), 933
- hasphon() (pynlpl.formats.folia.BegindatetimeFeature method), 898
- hasphon() (pynlpl.formats.folia.Cell method), 104
- hasphon() (pynlpl.formats.folia.Chunk method), 521
- hasphon() (pynlpl.formats.folia.ChunkingLayer method), 649
- hasphon() (pynlpl.formats.folia.CoreferenceChain method), 532
- hasphon() (pynlpl.formats.folia.CoreferenceLayer method), 661
- hasphon() (pynlpl.formats.folia.CoreferenceLink method), 767
- hasphon() (pynlpl.formats.folia.Correction method), 947
- hasphon() (pynlpl.formats.folia.Current method), 956 hasphon() (pynlpl.formats.folia.Definition method), 117
- hasphon() (pynlpl.formats.folia.DependenciesLayer method), 673
- hasphon() (pynlpl.formats.folia.Dependency method), 544
- hasphon() (pynlpl.formats.folia.DependencyDependent method), 779
- hasphon() (pynlpl.formats.folia.Description method), 1034
- hasphon() (pynlpl.formats.folia.Division method), 130 hasphon() (pynlpl.formats.folia.DomainAnnotation method), 431
- hasphon() (pynlpl.formats.folia.EnddatetimeFeature method), 909
- hasphon() (pynlpl.formats.folia.EntitiesLayer method), 685
- hasphon() (pynlpl.formats.folia.Entity method), 555
- hasphon() (pynlpl.formats.folia.Entry method), 143
- hasphon() (pynlpl.formats.folia.ErrorDetection method), 967
- hasphon() (pynlpl.formats.folia.Event method), 156
- hasphon() (pynlpl.formats.folia.Example method), 169

- $hasphon()\ (pynlpl.formats.folia.Feature\ method),\ 865$
- $hasphon()\ (pynlpl.formats.folia. Figure\ method),\ 182$
- hasphon() (pynlpl.formats.folia.Gap method), 193
- hasphon() (pynlpl.formats.folia.Head method), 206
- hasphon() (pynlpl.formats.folia.Headspan method), 790 hasphon() (pynlpl.formats.folia.LangAnnotation
  - method), 453
- hasphon() (pynlpl.formats.folia.LemmaAnnotation method), 464
- hasphon() (pynlpl.formats.folia.Linebreak method), 219
- hasphon() (pynlpl.formats.folia.List method), 232
- $hasphon()\ (pynlpl.formats.folia.ListItem\ method),\ 245$
- hasphon() (pynlpl.formats.folia.Metric method), 1045
- hasphon() (pynlpl.formats.folia.New method), 978
- hasphon() (pynlpl.formats.folia.Note method), 258
- hasphon() (pynlpl.formats.folia.Observation method), 567
- hasphon() (pynlpl.formats.folia.ObservationLayer method), 697
- hasphon() (pynlpl.formats.folia.Original method), 989
- $has phon ()\ (pynlpl. formats. folia. Paragraph\ method),\ 271$
- hasphon() (pynlpl.formats.folia.Part method), 284
- $\begin{array}{c} hasphon() \quad (pynlpl.formats.folia.PhonContent \quad method), \\ 508 \end{array}$
- hasphon() (pynlpl.formats.folia.PosAnnotation method), 442
- hasphon() (pynlpl.formats.folia.Predicate method), 579
- hasphon() (pynlpl.formats.folia.Quote method), 297
- hasphon() (pynlpl.formats.folia.Reference method), 310
- hasphon() (pynlpl.formats.folia.Row method), 323
- hasphon() (pynlpl.formats.folia.SemanticRole method), 625
- hasphon() (pynlpl.formats.folia.SemanticRolesLayer method), 744
- hasphon() (pynlpl.formats.folia.SenseAnnotation method), 475
- hasphon() (pynlpl.formats.folia.Sentence method), 337
- hasphon() (pynlpl.formats.folia.Sentiment method), 590 hasphon() (pynlpl.formats.folia.SentimentLayer method),
- 708 hasphon() (pynlpl.formats.folia.Statement method), 602
- hasphon() (pynlpl.formats.folia.StatementLayer method),
- hasphon() (pynlpl.formats.folia.SubjectivityAnnotation method), 486
- hasphon() (pynlpl.formats.folia.Suggestion method), 1000
- hasphon() (pynlpl.formats.folia.SynsetFeature method),
- hasphon() (pynlpl.formats.folia.SyntacticUnit method),
- hasphon() (pynlpl.formats.folia.SyntaxLayer method), 732
- hasphon() (pynlpl.formats.folia.Table method), 351

hasphon() (pynlpl.formats.folia.TableHead method), 377 hasphon() (pynlpl.formats.folia.Term method), 364 hasphon() (pynlpl.formats.folia.Text method), 390 hasphon() (pynlpl.formats.folia.TextContent method), hasphon() (pynlpl.formats.folia.TextMarkupCorrection method), 843 hasphon() (pynlpl.formats.folia.TextMarkupError method), 853 hasphon() (pynlpl.formats.folia.TextMarkupGap method), 811 (pynlpl.formats.folia.TextMarkupString hasphon() method), 821 (pynlpl.formats.folia.TextMarkupStyle hasphon() method), 832 hasphon() (pynlpl.formats.folia.TimeSegment method), 637 hasphon() (pynlpl.formats.folia.TimingLayer method), hasphon() (pynlpl.formats.folia.Whitespace method), 403 hasphon() (pynlpl.formats.folia.Word method), 417 (pynlpl.formats.folia.Correction hassuggestions() method), 947 hastext() (pynlpl.formats.folia.AbstractAnnotationLayer method), 78 hastext() (pynlpl.formats.folia.AbstractElement method), (pynlpl.formats.folia.AbstractSpanAnnotation hastext() method), 55 hastext() (pynlpl.formats.folia.AbstractStructureElement method), 41 hastext() (pynlpl.formats.folia.AbstractTextMarkup method), 89 hastext() (pynlpl.formats.folia.AbstractTokenAnnotation method), 66 hastext() (pynlpl.formats.folia.ActorFeature method), 888 hastext() (pynlpl.formats.folia.Alignment method), 1011 hastext() (pynlpl.formats.folia.AlignReference method), 1023 hastext() (pynlpl.formats.folia.Alternative method), 923 (pynlpl.formats.folia.AlternativeLayers hastext() method), 934 (pynlpl.formats.folia.BegindatetimeFeature hastext() method), 899 hastext() (pynlpl.formats.folia.Cell method), 104 hastext() (pynlpl.formats.folia.Chunk method), 521 hastext() (pynlpl.formats.folia.ChunkingLayer method), 650 (pynlpl.formats.folia.CoreferenceChain hastext() method), 532

(pynlpl.formats.folia.CoreferenceLayer

hastext() (pynlpl.formats.folia.CoreferenceLink method),

hastext()

method), 662

767

```
hastext() (pynlpl.formats.folia.Correction method), 947
hastext() (pynlpl.formats.folia.Current method), 956
hastext() (pynlpl.formats.folia.Definition method), 117
hastext()
                (pynlpl.formats.folia.DependenciesLayer
         method), 673
hastext() (pynlpl.formats.folia.Dependency method), 544
hastext()
            (pynlpl.formats.folia.DependencyDependent
         method), 779
hastext() (pynlpl.formats.folia.Description method), 1034
hastext() (pynlpl.formats.folia.Division method), 130
hastext()
                (pynlpl.formats.folia.DomainAnnotation
         method), 431
               (pynlpl.formats.folia.EnddatetimeFeature
hastext()
         method), 910
hastext()
          (pynlpl.formats.folia.EntitiesLayer method),
hastext() (pynlpl.formats.folia.Entity method), 556
hastext() (pynlpl.formats.folia.Entry method), 143
hastext() (pynlpl.formats.folia.ErrorDetection method),
hastext() (pynlpl.formats.folia.Event method), 156
hastext() (pynlpl.formats.folia.Example method), 169
hastext() (pynlpl.formats.folia.Feature method), 865
hastext() (pynlpl.formats.folia.Figure method), 182
hastext() (pynlpl.formats.folia.Gap method), 194
hastext() (pynlpl.formats.folia.Head method), 206
hastext() (pynlpl.formats.folia.Headspan method), 791
hastext() (pynlpl.formats.folia.LangAnnotation method),
                 (pynlpl.formats.folia.LemmaAnnotation
hastext()
          method), 464
hastext() (pynlpl.formats.folia.Linebreak method), 219
hastext() (pynlpl.formats.folia.List method), 232
hastext() (pynlpl.formats.folia.ListItem method), 245
hastext() (pynlpl.formats.folia.Metric method), 1045
hastext() (pynlpl.formats.folia.New method), 978
hastext() (pynlpl.formats.folia.Note method), 258
hastext() (pynlpl.formats.folia.Observation method), 567
                  (pynlpl.formats.folia.ObservationLayer
hastext()
         method), 697
hastext() (pynlpl.formats.folia.Original method), 989
hastext() (pynlpl.formats.folia.Paragraph method), 271
hastext() (pynlpl.formats.folia.Part method), 284
hastext() (pynlpl.formats.folia.PhonContent method), 508
hastext() (pynlpl.formats.folia.PosAnnotation method),
          442
hastext() (pynlpl.formats.folia.Predicate method), 579
hastext() (pynlpl.formats.folia.Quote method), 297
hastext() (pynlpl.formats.folia.Reference method), 310
hastext() (pynlpl.formats.folia.Row method), 323
hastext() (pynlpl.formats.folia.SemanticRole method),
```

(pynlpl.formats.folia.SemanticRolesLayer

1118 Index

hastext()

method), 744

hastext() (pynlpl.formats.folia.SenseAnnotation method),	incorrection() (pynlpl.formats.folia.AbstractTextMarkup	
475 hastext() (pynlpl.formats.folia.Sentence method), 337	method), 89 incorrection() (pynlpl.formats.folia.AbstractTokenAnnotation	
hastext() (pynlpl.formats.folia.Sentiment method), 591	method), 66	
hastext() (pynlpl.formats.folia.SentimentLayer method),	incorrection() (pynlpl.formats.folia.ActorFeature method), 888	
hastext() (pynlpl.formats.folia.Statement method), 602	incorrection() (pynlpl.formats.folia.Alignment method),	
hastext() (pynlpl.formats.folia.StatementLayer method), 720	incorrection() (pynlpl.formats.folia.AlignReference	
hastext() (pynlpl.formats.folia.SubjectivityAnnotation	method), 1023	
method), 486	incorrection() (pynlpl.formats.folia.Alternative method),	
hastext() (pynlpl.formats.folia.Suggestion method), 1000	923	
hastext() (pynlpl.formats.folia.SynsetFeature method), 876	incorrection() (pynlpl.formats.folia.AlternativeLayers method), 934	
hastext() (pynlpl.formats.folia.SyntacticUnit method), 614	incorrection() (pynlpl.formats.folia.BegindatetimeFeature method), 899	
hastext() (pynlpl.formats.folia.SyntaxLayer method), 732	incorrection() (pynlpl.formats.folia.Cell method), 105	
hastext() (pynlpl.formats.folia.Table method), 351	incorrection() (pynlpl.formats.folia.Chunk method), 521	
hastext() (pynlpl.formats.folia.TableHead method), 377	incorrection() (pynlpl.formats.folia.ChunkingLayer	
hastext() (pynlpl.formats.folia.Term method), 364	method), 650	
hastext() (pynlpl.formats.folia.Text method), 390 hastext() (pynlpl.formats.folia.TextContent method), 498	incorrection() (pynlpl.formats.folia.CoreferenceChain	
hastext() (pynlpl.formats.folia.TextContent method), 498 hastext() (pynlpl.formats.folia.TextMarkupCorrection	method), 533 incorrection() (pynlpl.formats.folia.CoreferenceLayer	
method), 843	method), 662	
hastext() (pynlpl.formats.folia.TextMarkupError	incorrection() (pynlpl.formats.folia.CoreferenceLink	
method), 854	method), 768	
hastext() (pynlpl.formats.folia.TextMarkupGap method), 811	incorrection() (pynlpl.formats.folia.Correction method), 947	
hastext() (pynlpl.formats.folia.TextMarkupString	incorrection() (pynlpl.formats.folia.Current method), 956	
method), 822 hastext() (pynlpl.formats.folia.TextMarkupStyle method),	incorrection() (pynlpl.formats.folia.Definition method), 118	
832	incorrection() (pynlpl.formats.folia.DependenciesLayer	
hastext() (pynlpl.formats.folia.TimeSegment method),	method), 674	
637	incorrection() (pynlpl.formats.folia.Dependency method),	
hastext() (pynlpl.formats.folia.TimingLayer method), 756	545	
hastext() (pynlpl.formats.folia.Whitespace method), 403 hastext() (pynlpl.formats.folia.Word method), 418	incorrection() (pynlpl.formats.folia.DependencyDependent method), 779	
Head (class in pynlpl.formats.folia), 200	incorrection() (pynlpl.formats.folia.Description method),	
head() (pynlpl.formats.folia.Dependency method), 544	1034	
head() (pynlpl.formats.folia.Division method), 130	incorrection() (pynlpl.formats.folia.Division method),	
Headspan (class in pynlpl.formats.folia), 785	131	
HiddenMarkovModel (class in pynlpl.statistics), 1063	incorrection() (pynlpl.formats.folia.DomainAnnotation	
HillClimbingSearch (class in pynlpl.search), 1060	method), 431	
histogram() (in module pynlpl.statistics), 1064	incorrection() (pynlpl.formats.folia.EnddatetimeFeature method), 910	
1	incorrection() (pynlpl.formats.folia.EntitiesLayer	
incorrection() (pynlpl.formats.folia.AbstractAnnotationLay	ver method), 686	
method), 78	incorrection() (pynlpl.formats.folia.Entity method), 556	
incorrection() (pynlpl.formats.folia.AbstractElement method), 29	incorrection() (pynlpl.formats.folia.Entry method), 143 incorrection() (pynlpl.formats.folia.ErrorDetection	
incorrection() (pynlpl.formats.folia.AbstractSpanAnnotation method), 967		
method), 55	incorrection() (pynlpl.formats.folia.Event method), 156	
incorrection() (pynlpl.formats.folia.AbstractStructureElem method), 41	entrocorrection() (pynlpl.formats.folia.Example method),	

incorrection() (pynlpl.formats.folia.Feature method), 865 incorrection() (pynlpl.formats.folia.Figure method), 183 incorrection() (pynlpl.formats.folia.Gap method), 194 incorrection() (pynlpl.formats.folia.Head method), 207 incorrection() (pynlpl.formats.folia.Headspan method), incorrection() (pynlpl.formats.folia.LangAnnotation method), 453 incorrection() (pynlpl.formats.folia.LemmaAnnotation method), 464 incorrection() (pynlpl.formats.folia.Linebreak method), 220 incorrection() (pynlpl.formats.folia.List method), 232 (pynlpl.formats.folia.ListItem method), incorrection() 245 incorrection() (pynlpl.formats.folia.Metric method), 1045 incorrection() (pynlpl.formats.folia.New method), 978 incorrection() (pynlpl.formats.folia.Note method), 258 incorrection() (pynlpl.formats.folia.Observation method), 568 incorrection() (pynlpl.formats.folia.ObservationLayer method), 697 (pynlpl.formats.folia.Original method), incorrection() incorrection() (pynlpl.formats.folia.Paragraph method), incorrection() (pynlpl.formats.folia.Part method), 284 incorrection() (pynlpl.formats.folia.PhonContent method), 509 (pynlpl.formats.folia.PosAnnotation incorrection() method), 442 incorrection() (pynlpl.formats.folia.Predicate method), incorrection() (pynlpl.formats.folia.Quote method), 297 incorrection() (pynlpl.formats.folia.Reference method), incorrection() (pynlpl.formats.folia.Row method), 323 incorrection() (pynlpl.formats.folia.SemanticRole method), 626 incorrection() (pynlpl.formats.folia.SemanticRolesLayer method), 744 incorrection() (pynlpl.formats.folia.SenseAnnotation method), 475 (pynlpl.formats.folia.Sentence method), incorrection() incorrection() (pynlpl.formats.folia.Sentiment method), 591 incorrection() (pynlpl.formats.folia.SentimentLayer method), 709 incorrection() (pynlpl.formats.folia.Statement method), 603 incorrection() (pynlpl.formats.folia.StatementLayer

method), 721

method), 486 incorrection() (pynlpl.formats.folia.Suggestion method), 1000 (pynlpl.formats.folia.SynsetFeature incorrection() method), 877 incorrection() (pynlpl.formats.folia.SyntacticUnit method), 614 (pynlpl.formats.folia.SyntaxLayer incorrection() method), 733 incorrection() (pynlpl.formats.folia.Table method), 351 incorrection() (pynlpl.formats.folia.TableHead method), incorrection() (pynlpl.formats.folia.Term method), 364 incorrection() (pynlpl.formats.folia.Text method), 390 incorrection() (pynlpl.formats.folia.TextContent method), incorrection() (pynlpl.formats.folia.TextMarkupCorrection method), 843 incorrection() (pynlpl.formats.folia.TextMarkupError method), 854 incorrection() (pynlpl.formats.folia.TextMarkupGap method), 811 incorrection() (pynlpl.formats.folia.TextMarkupString method), 822 (pynlpl.formats.folia.TextMarkupStyle incorrection() method), 833 incorrection() (pynlpl.formats.folia.TimeSegment method), 638 (pynlpl.formats.folia.TimingLayer incorrection() method), 756 incorrection() (pynlpl.formats.folia.Whitespace method), incorrection() (pynlpl.formats.folia.Word method), 418 information() (pynlpl.statistics.Distribution method), 1062 initdoc() (pynlpl.formats.folia.Reader method), 805 insert() (pynlpl.formats.folia.AbstractAnnotationLayer method), 78 insert() (pynlpl.formats.folia.AbstractElement method), insert() (pynlpl.formats.folia.AbstractSpanAnnotation method), 55 (pynlpl.formats.folia.AbstractStructureElement insert() method), 42 (pynlpl.formats.folia.AbstractTextMarkup insert() method), 89 (pynlpl.formats.folia.AbstractTokenAnnotation insert() method), 66 insert() (pynlpl.formats.folia.ActorFeature method), 888 insert() (pynlpl.formats.folia.Alignment method), 1012 insert() (pynlpl.formats.folia.AlignReference method),

insert() (pynlpl.formats.folia.Alternative method), 923

1120 Index

incorrection() (pynlpl.formats.folia.SubjectivityAnnotation insert() (pynlpl.formats.folia.AlternativeLayers method),

024	
934	insert() (pynlpl.formats.folia.Part method), 285
insert() (pynlpl.formats.folia.BegindatetimeFeature	insert() (pynlpl.formats.folia.PhonContent method), 509
method), 899	insert() (pynlpl.formats.folia.PosAnnotation method), 443
insert() (pynlpl.formats.folia.Cell method), 105 insert() (pynlpl.formats.folia.Chunk method), 521	insert() (pynlpl.formats.folia.Predicate method), 579
insert() (pynlpl.formats.folia.ChunkingLayer method),	insert() (pynlpl.formats.folia.Predicate method), 379
650	insert() (pynlpl.formats.folia.Reference method), 311
insert() (pynlpl.formats.folia.CoreferenceChain method),	insert() (pynlpl.formats.folia.Row method), 324
533	insert() (pynlpl.formats.folia.Row method), 324 insert() (pynlpl.formats.folia.SemanticRole method), 626
insert() (pynlpl.formats.folia.CoreferenceLayer method),	insert() (pynlpl.formats.folia.SemanticRolesLayer
662	method), 744
insert() (pynlpl.formats.folia.CoreferenceLink method), 768	insert() (pynlpl.formats.folia.SenseAnnotation method), 476
insert() (pynlpl.formats.folia.Correction method), 947	insert() (pynlpl.formats.folia.Sentence method), 338
insert() (pynlpl.formats.folia.Current method), 956	insert() (pynlpl.formats.folia.Sentiment method), 591
insert() (pynlpl.formats.folia.Definition method), 118	insert() (pynlpl.formats.folia.SentimentLayer method),
insert() (pynlpl.formats.folia.DependenciesLayer	709
method), 674	insert() (pynlpl.formats.folia.Statement method), 603
insert() (pynlpl.formats.folia.Dependency method), 545	insert() (pynlpl.formats.folia.StatementLayer method),
insert() (pynlpl.formats.folia.DependencyDependent	721
method), 779	insert() (pynlpl.formats.folia.SubjectivityAnnotation
insert() (pynlpl.formats.folia.Description method), 1034	method), 487
insert() (pynlpl.formats.folia.Division method), 131	insert() (pynlpl.formats.folia.Suggestion method), 1001
insert() (pynlpl.formats.folia.DomainAnnotation method), 432	insert() (pynlpl.formats.folia.SynsetFeature method), 877 insert() (pynlpl.formats.folia.SyntacticUnit method), 614
insert() (pynlpl.formats.folia.EnddatetimeFeature	insert() (pynlpl.formats.folia.Syntacticonit method), 733
method), 910	insert() (pynlpl.formats.folia.Table method), 351
insert() (pynlpl.formats.folia.EntitiesLayer method), 686	insert() (pynlpl.formats.folia.TableHead method), 377
insert() (pynlpl.formats.folia.Entity method), 556	insert() (pynlpl.formats.folia.Term method), 364
insert() (pynlpl.formats.folia.Entry method), 144	insert() (pynlpl.formats.folia.Text method), 390
insert() (pynlpl.formats.folia.ErrorDetection method),	insert() (pynlpl.formats.folia.TextContent method), 498
968	insert() (pynlpl.formats.folia.TextMarkupCorrection
insert() (pynlpl.formats.folia.Event method), 157	method), 843
insert() (pynlpl.formats.folia.Example method), 170	insert() (pynlpl.formats.folia.TextMarkupError method),
insert() (pynlpl.formats.folia.Feature method), 866	854
insert() (pynlpl.formats.folia.Figure method), 183	insert() (pynlpl.formats.folia.TextMarkupGap method),
insert() (pynlpl.formats.folia.Gap method), 194	811
insert() (pynlpl.formats.folia.Head method), 207	insert() (pynlpl.formats.folia.TextMarkupString method),
insert() (pynlpl.formats.folia.Headspan method), 791	822
insert() (pynlpl.formats.folia.LangAnnotation method), 454	insert() (pynlpl.formats.folia.TextMarkupStyle method), 833
insert() (pynlpl.formats.folia.LemmaAnnotation method),	insert() (pynlpl.formats.folia.TimeSegment method), 638
465	insert() (pynlpl.formats.folia.TimingLayer method), 756
insert() (pynlpl.formats.folia.Linebreak method), 220	insert() (pynlpl.formats.folia.Whitespace method), 403
insert() (pynlpl.formats.folia.List method), 233	insert() (pynlpl.formats.folia.Word method), 418
insert() (pynlpl.formats.folia.ListItem method), 246	insertword() (pynlpl.formats.folia.Sentence method), 338
insert() (pynlpl.formats.folia.Metric method), 1046	insertwordleft() (pynlpl.formats.folia.Sentence method),
insert() (pynlpl.formats.folia.New method), 979	338
insert() (pynlpl.formats.folia.Note method), 259	intersect() (pynlpl.formats.giza.GizaSentenceAlignment
insert() (pynlpl.formats.folia.Observation method), 568	method), 1053
insert() (pynlpl.formats.folia.ObservationLayer method), 697	IntersectionAlignment (class in pynlpl.formats.giza), 1053
insert() (pynlpl.formats.folia.Original method), 990	
insert() (pynlpl.formats.folia.Paragraph method), 272	InvalidFeatureException, 1053 InvalidTagException, 1053

is end of sentence() (in module pynlpl.textprocessors), items() (pynlpl.formats.folia.ErrorDetection method), 1066 968 isstring() (in module pynlpl.common), 3 items() (pynlpl.formats.folia.Event method), 157 items() (pynlpl.datatypes.PatternMap method), 5 items() (pynlpl.formats.folia.Example method), 170 items() (pynlpl.datatypes.Trie method), 6 items() (pynlpl.formats.folia.Feature method), 866 items() (pynlpl.formats.folia.AbstractAnnotationLayer items() (pynlpl.formats.folia.Figure method), 183 method), 78 items() (pynlpl.formats.folia.Gap method), 194 items() (pynlpl.formats.folia.Head method), 207 items() (pynlpl.formats.folia.AbstractElement method), items() (pynlpl.formats.folia.Headspan method), 791 (pynlpl.formats.folia.AbstractSpanAnnotation items() (pynlpl.formats.folia.LangAnnotation method), items() method), 55 (pynlpl.formats.folia.AbstractStructureElement items() (pynlpl.formats.folia.LemmaAnnotation method), items() method), 42 items() (pynlpl.formats.folia.AbstractTextMarkup items() (pynlpl.formats.folia.Linebreak method), 220 method), 89 items() (pynlpl.formats.folia.List method), 233 items() (pynlpl.formats.folia.AbstractTokenAnnotation items() (pynlpl.formats.folia.ListItem method), 246 method), 66 items() (pynlpl.formats.folia.Metric method), 1046 items() (pynlpl.formats.folia.ActorFeature method), 888 items() (pynlpl.formats.folia.New method), 979 items() (pynlpl.formats.folia.Alignment method), 1012 items() (pynlpl.formats.folia.Note method), 259 items() (pynlpl.formats.folia.AlignReference method), items() (pynlpl.formats.folia.Observation method), 568 1023 items() (pynlpl.formats.folia.ObservationLayer method), items() (pynlpl.formats.folia.Alternative method), 923 items() (pynlpl.formats.folia.AlternativeLayers method), items() (pynlpl.formats.folia.Original method), 990 items() (pynlpl.formats.folia.Paragraph method), 272 (pynlpl.formats.folia.BegindatetimeFeature items() (pynlpl.formats.folia.Part method), 285 items() method), 899 items() (pynlpl.formats.folia.PhonContent method), 509 items() (pynlpl.formats.folia.Cell method), 105 items() (pynlpl.formats.folia.PosAnnotation method), 443 items() (pynlpl.formats.folia.Chunk method), 521 items() (pynlpl.formats.folia.Predicate method), 579 items() (pynlpl.formats.folia.ChunkingLayer method), items() (pynlpl.formats.folia.Quote method), 298 items() (pynlpl.formats.folia.Reference method), 311 items() (pynlpl.formats.folia.CoreferenceChain method), items() (pynlpl.formats.folia.Row method), 324 533 items() (pynlpl.formats.folia.SemanticRole method), 626 items() (pynlpl.formats.folia.CoreferenceLayer method), (pynlpl.formats.folia.SemanticRolesLayer items() method), 744 items() (pynlpl.formats.folia.CoreferenceLink method), items() (pynlpl.formats.folia.SenseAnnotation method), items() (pynlpl.formats.folia.Correction method), 947 items() (pynlpl.formats.folia.Sentence method), 338 items() (pynlpl.formats.folia.Current method), 956 items() (pynlpl.formats.folia.Sentiment method), 591 items() (pynlpl.formats.folia.Definition method), 118 items() (pynlpl.formats.folia.SentimentLayer method), (pynlpl.formats.folia.DependenciesLayer items() method), 674 items() (pynlpl.formats.folia.Statement method), 603 items() (pynlpl.formats.folia.Dependency method), 545 items() (pynlpl.formats.folia.StatementLayer method), (pynlpl.formats.folia.DependencyDependent items() method), 779 items() (pynlpl.formats.folia.SubjectivityAnnotation items() (pynlpl.formats.folia.Description method), 1034 method), 487 items() (pynlpl.formats.folia.Division method), 131 items() (pynlpl.formats.folia.Suggestion method), 1001 items() (pynlpl.formats.folia.Document method), 19 items() (pynlpl.formats.folia.SynsetFeature method), 877 items() (pynlpl.formats.folia.DomainAnnotation items() (pynlpl.formats.folia.SyntacticUnit method), 614 items() (pynlpl.formats.folia.SyntaxLayer method), 733 method), 432 (pynlpl.formats.folia.EnddatetimeFeature items() (pynlpl.formats.folia.Table method), 351 items() method), 910 items() (pynlpl.formats.folia.TableHead method), 377 items() (pynlpl.formats.folia.EntitiesLayer method), 686 items() (pynlpl.formats.folia.Term method), 364 items() (pynlpl.formats.folia.Entity method), 556 items() (pynlpl.formats.folia.Text method), 390 items() (pynlpl.formats.folia.Entry method), 144 items() (pynlpl.formats.folia.TextContent method), 498

items() (pynlpl.formats.folia.TextMarkupCorrection method), 843	json() (pynlpl.formats.folia.Dependency method), 545 json() (pynlpl.formats.folia.DependencyDependent
items() (pynlpl.formats.folia.TextMarkupError method), 854	method), 779 json() (pynlpl.formats.folia.Description method), 1035
items() (pynlpl.formats.folia.TextMarkupGap method),	json() (pynlpl.formats.folia.Division method), 131 json() (pynlpl.formats.folia.Document method), 19
items() (pynlpl.formats.folia.TextMarkupString method), 822	json() (pynlpl.formats.folia.DomainAnnotation method), 432
items() (pynlpl.formats.folia.TextMarkupStyle method), 833	json() (pynlpl.formats.folia.EnddatetimeFeature method), 910
items() (pynlpl.formats.folia.TimeSegment method), 638	json() (pynlpl.formats.folia.EntitiesLayer method), 686
items() (pynlpl.formats.folia.TimingLayer method), 756	json() (pynlpl.formats.folia.Entity method), 556
items() (pynlpl.formats.folia.Whitespace method), 403	json() (pynlpl.formats.folia.Entry method), 144
items() (pynlpl.formats.folia.Word method), 418	json() (pynlpl.formats.folia.ErrorDetection method), 968
items() (pynlpl.statistics.Distribution method), 1062	json() (pynlpl.formats.folia.Event method), 157
items() (pynlpl.statistics.FrequencyList method), 1063	json() (pynlpl.formats.folia.Example method), 170
IterativeDeepening (class in pynlpl.search), 1060	json() (pynlpl.formats.folia.Feature method), 866
iterbytes() (pynlpl.datatypes.Pattern method), 5	json() (pynlpl.formats.folia.Figure method), 183
	json() (pynlpl.formats.folia.Gap method), 194
J	json() (pynlpl.formats.folia.Head method), 207
json() (pynlpl.formats.folia.AbstractAnnotationLayer	json() (pynlpl.formats.folia.Headspan method), 791
method), 78	json() (pynlpl.formats.folia.LangAnnotation method),
json() (pynlpl.formats.folia.AbstractElement method), 29	454
json() (pynlpl.formats.folia.AbstractSpanAnnotation method), 55	json() (pynlpl.formats.folia.LemmaAnnotation method), 465
json() (pynlpl.formats.folia.AbstractStructureElement	json() (pynlpl.formats.folia.Linebreak method), 220
method), 42	json() (pynlpl.formats.folia.List method), 233
json() (pynlpl.formats.folia.AbstractTextMarkup	json() (pynlpl.formats.folia.ListItem method), 246
method), 89	json() (pynlpl.formats.folia.Metric method), 1046
json() (pynlpl.formats.folia.AbstractTokenAnnotation	json() (pynlpl.formats.folia.New method), 979
method), 66	json() (pynlpl.formats.folia.Note method), 259
json() (pynlpl.formats.folia.ActorFeature method), 888	json() (pynlpl.formats.folia.Observation method), 568
json() (pynlpl.formats.folia.Alignment method), 1012	json() (pynlpl.formats.folia.ObservationLayer method),
json() (pynlpl.formats.folia.AlignReference method),	697
1023	json() (pynlpl.formats.folia.Original method), 990
json() (pynlpl.formats.folia.Alternative method), 923	json() (pynlpl.formats.folia.Paragraph method), 272
json() (pynlpl.formats.folia.AlternativeLayers method),	json() (pynlpl.formats.folia.Part method), 285
934	json() (pynlpl.formats.folia.PhonContent method), 509
json() (pynlpl.formats.folia.BegindatetimeFeature	json() (pynlpl.formats.folia.PosAnnotation method), 443
method), 899	json() (pynlpl.formats.folia.Predicate method), 579
json() (pynlpl.formats.folia.Cell method), 105	json() (pynlpl.formats.folia.Quote method), 298
json() (pynlpl.formats.folia.Chunk method), 521	json() (pynlpl.formats.folia.Reference method), 311
json() (pynlpl.formats.folia.ChunkingLayer method), 650	json() (pynlpl.formats.folia.Row method), 324
json() (pynlpl.formats.folia.CoreferenceChain method), 533	json() (pynlpl.formats.folia.SemanticRole method), 626 json() (pynlpl.formats.folia.SemanticRolesLayer
json() (pynlpl.formats.folia.CoreferenceLayer method), 662	method), 744 json() (pynlpl.formats.folia.SenseAnnotation method),
json() (pynlpl.formats.folia.CoreferenceLink method),	476
768	json() (pynlpl.formats.folia.Sentence method), 338
json() (pynlpl.formats.folia.Correction method), 947	json() (pynlpl.formats.folia.Sentiment method), 591
json() (pynlpl.formats.folia.Current method), 956	json() (pynlpl.formats.folia.SentimentLayer method), 709
json() (pynlpl.formats.folia.Definition method), 118	json() (pynlpl.formats.folia.Statement method), 603
json() (pynlpl.formats.folia.DependenciesLayer method), 674	json() (pynlpl.formats.folia.StatementLayer method), 721

json() (pynlpl.formats.folia.SubjectivityAnnotation method), 487	LABEL (pynlpl.formats.folia.DomainAnnotation attribute), 428
json() (pynlpl.formats.folia.Suggestion method), 1001 json() (pynlpl.formats.folia.SynsetFeature method), 877	LABEL (pynlpl.formats.folia.EnddatetimeFeature attribute), 907
json() (pynlpl.formats.folia.SyntacticUnit method), 614	LABEL (pynlpl.formats.folia.Entity attribute), 553
json() (pynlpl.formats.folia.SyntaxLayer method), 733	LABEL (pynlpl.formats.folia.Entry attribute), 333
json() (pynlpl.formats.folia.Table method), 351	LABEL (pynlpl.formats.folia.ErrorDetection attribute),
json() (pynlpl.formats.folia.TableHead method), 377	964
json() (pynlpl.formats.folia.Term method), 364	LABEL (pynlpl.formats.folia.Event attribute), 152
json() (pynlpl.formats.folia.Text method), 390	LABEL (pynlpl.formats.folia.Example attribute), 165
json() (pynlpl.formats.folia.TextContent method), 498	LABEL (pynlpl.formats.folia.Feature attribute), 862
json() (pynlpl.formats.folia.TextMarkupCorrection	LABEL (pynlpl.formats.folia.Figure attribute), 178
method), 843	LABEL (pynlpl.formats.folia.Gap attribute), 191
json() (pynlpl.formats.folia.TextMarkupError method),	LABEL (pynlpl.formats.folia.Head attribute), 202
854	LABEL (pynlpl.formats.folia.Headspan attribute), 788
json() (pynlpl.formats.folia.TextMarkupGap method), 811	LABEL (pynlpl.formats.folia.LangAnnotation attribute), 450
json() (pynlpl.formats.folia.TextMarkupString method), 822	LABEL (pynlpl.formats.folia.LemmaAnnotation attribute), 461
json() (pynlpl.formats.folia.TextMarkupStyle method), 833	LABEL (pynlpl.formats.folia.Linebreak attribute), 215 LABEL (pynlpl.formats.folia.List attribute), 228
json() (pynlpl.formats.folia.TimeSegment method), 638	LABEL (pynlpl.formats.folia.ListItem attribute), 241
json() (pynlpl.formats.folia.TimingLayer method), 756	LABEL (pynlpl.formats.folia.Metric attribute), 1042
json() (pynlpl.formats.folia.Whitespace method), 403	LABEL (pynlpl.formats.folia.Note attribute), 254
json() (pynlpl.formats.folia.Word method), 418	LABEL (pynlpl.formats.folia.Observation attribute), 564
jsondeclarations() (pynlpl.formats.folia.Document	LABEL (pynlpl.formats.folia.Paragraph attribute), 267
method), 19	LABEL (pynlpl.formats.folia.Part attribute), 280
	LABEL (pynlpl.formats.folia.PhonContent attribute), 505
K	LABEL (pynlpl.formats.folia.PosAnnotation attribute),
keys() (pynlpl.statistics.Distribution method), 1062	439
, , , , , , , , , , , , , , , , , , ,	LABEL (pynlpl.formats.folia.Predicate attribute), 576
L	LABEL (pynlpl.formats.folia.Quote attribute), 293
LABEL (pynlpl.formats.folia.ActorFeature attribute),	LABEL (pynlpl.formats.folia.Reference attribute), 306
885	LABEL (pynlpl.formats.folia.Row attribute), 319
LABEL (pynlpl.formats.folia.Alignment attribute), 1009	LABEL (pynlpl.formats.folia.SemanticRole attribute),
LABEL (pynlpl.formats.folia.Alternative attribute), 919	623
LABEL (pynlpl.formats.folia.AlternativeLayers attribute), 931	LABEL (pynlpl.formats.folia.SenseAnnotation attribute), 472
LABEL (pynlpl.formats.folia.BegindatetimeFeature at-	LABEL (pynlpl.formats.folia.Sentence attribute), 332
tribute), 896	LABEL (pynlpl.formats.folia.Sentiment attribute), 588
LABEL (pynlpl.formats.folia.Cell attribute), 100	LABEL (pynlpl.formats.folia.Statement attribute), 599
LABEL (pynlpl.formats.folia.Chunk attribute), 518	LABEL (pynlpl.formats.folia.SubjectivityAnnotation at-
LABEL (pynlpl.formats.folia.CoreferenceChain at-	tribute), 483
tribute), 529	LABEL (pynlpl.formats.folia.SynsetFeature attribute), 873
LABEL (pynlpl.formats.folia.CoreferenceLink attribute), 764	LABEL (pynlpl.formats.folia.SyntacticUnit attribute), 611
LABEL (pynlpl.formats.folia.Correction attribute), 944	LABEL (pynlpl.formats.folia.Table attribute), 347
LABEL (pynlpl.formats.folia.Definition attribute), 113	LABEL (pynlpl.formats.folia.TableHead attribute), 373
LABEL (pynlpl.formats.folia.Dependency attribute), 541	LABEL (pynlpl.formats.folia.Term attribute), 360
LABEL (pynlpl.formats.folia.DependencyDependent at-	LABEL (pynlpl.formats.folia.Text attribute), 386
tribute), 776	LABEL (pynlpl.formats.folia.Text attribute), 495
LABEL (pynlpl.formats.folia.Description attribute), 1031	LABEL (pynlpl.formats.folia.TimeSegment attribute),
LABEL (pynlpl.formats.folia.Division attribute), 126	634

LABEL (pynlpl.formats.folia.Whitespace attribute), 399	left context ()  (pynlpl. formats. folia. Begind at etime Feature
LABEL (pynlpl.formats.folia.Word attribute), 412	method), 899
LangAnnotation (class in pynlpl.formats.folia), 448	leftcontext() (pynlpl.formats.folia.Cell method), 105
language() (pynlpl.formats.folia.Document method), 19	leftcontext() (pynlpl.formats.folia.Chunk method), 522
layers() (pynlpl.formats.folia.AbstractStructureElement method), 42	leftcontext() (pynlpl.formats.folia.ChunkingLayer method), 650
layers() (pynlpl.formats.folia.Cell method), 105	leftcontext() (pynlpl.formats.folia.CoreferenceChain
layers() (pynlpl.formats.folia.Definition method), 118	method), 533
layers() (pynlpl.formats.folia.Division method), 131	leftcontext() (pynlpl.formats.folia.CoreferenceLayer
layers() (pynlpl.formats.folia.Entry method), 144	method), 662
layers() (pynlpl.formats.folia.Event method), 157	leftcontext() (pynlpl.formats.folia.CoreferenceLink
layers() (pynlpl.formats.folia.Example method), 170	method), 768
layers() (pynlpl.formats.folia.Figure method), 183	leftcontext() (pynlpl.formats.folia.Correction method),
layers() (pynlpl.formats.folia.Head method), 207	947
layers() (pynlpl.formats.folia.Linebreak method), 220	leftcontext() (pynlpl.formats.folia.Current method), 957
layers() (pynlpl.formats.folia.List method), 233	leftcontext() (pynlpl.formats.folia.Definition method),
layers() (pynlpl.formats.folia.ListItem method), 246	118
layers() (pynlpl.formats.folia.Note method), 259	leftcontext() (pynlpl.formats.folia.DependenciesLayer
layers() (pynlpl.formats.folia.Paragraph method), 272	method), 674
layers() (pynlpl.formats.folia.Part method), 285	leftcontext() (pynlpl.formats.folia.Dependency method),
layers() (pynlpl.formats.folia.Quote method), 298	545
layers() (pynlpl.formats.folia.Reference method), 311	leftcontext() (pynlpl.formats.folia.DependencyDependent
layers() (pynlpl.formats.folia.Row method), 324	method), 780
layers() (pynlpl.formats.folia.Sentence method), 338	leftcontext() (pynlpl.formats.folia.Description method),
layers() (pynlpl.formats.folia.Table method), 352	1035
layers() (pynlpl.formats.folia.TableHead method), 378	leftcontext() (pynlpl.formats.folia.Division method), 131
layers() (pynlpl.formats.folia.Term method), 365	leftcontext() (pynlpl.formats.folia.DomainAnnotation
layers() (pynlpl.formats.folia.Text method), 391	method), 432
layers() (pynlpl.formats.folia.Whitespace method), 404	leftcontext() (pynlpl.formats.folia.EnddatetimeFeature
layers() (pynlpl.formats.folia.Word method), 418	method), 910
leaf() (pynlpl.datatypes.Tree method), 6	leftcontext() (pynlpl.formats.folia.EntitiesLayer method),
leaf() (pynlpl.datatypes.Trie method), 6	686
leftcontext() (pynlpl.formats.folia.AbstractAnnotationLayer	er leftcontext() (pynlpl.formats.folia.Entity method), 556
method), 79	leftcontext() (pynlpl.formats.folia.Entry method), 144
leftcontext() (pynlpl.formats.folia.AbstractElement	leftcontext() (pynlpl.formats.folia.ErrorDetection
method), 29	method), 968
leftcontext() (pynlpl.formats.folia.AbstractSpanAnnotation	leftcontext() (pynlpl.formats.folia.Event method), 157
method), 56	leftcontext() (pynlpl.formats.folia.Example method), 170
leftcontext() (pynlpl.formats.folia.AbstractStructureEleme	ntleftcontext() (pynlpl.formats.folia.Feature method), 866
method), 42	leftcontext() (pynlpl.formats.folia.Figure method), 183
$leftcontext() \\ \hspace{0.5cm} (pynlpl.formats.folia. Abstract Text Markup \\$	leftcontext() (pynlpl.formats.folia.Gap method), 194
method), 89	leftcontext() (pynlpl.formats.folia.Head method), 207
leftcontext() (pynlpl.formats.folia.AbstractTokenAnnotation	onleftcontext() (pynlpl.formats.folia.Headspan method),
method), 67	791
leftcontext() (pynlpl.formats.folia.ActorFeature method),	leftcontext() (pynlpl.formats.folia.LangAnnotation
888	method), 454
leftcontext() (pynlpl.formats.folia.Alignment method),	leftcontext() (pynlpl.formats.folia.LemmaAnnotation
1012	method), 465
leftcontext() (pynlpl.formats.folia.AlignReference	leftcontext() (pynlpl.formats.folia.Linebreak method),
method), 1023	220
leftcontext() (pynlpl.formats.folia.Alternative method),	leftcontext() (pynlpl.formats.folia.List method), 233
923	leftcontext() (pynlpl.formats.folia.ListItem method), 246
leftcontext() (pynlpl.formats.folia.AlternativeLayers	leftcontext() (pynlpl.formats.folia.Metric method), 1046
method), 934	leftcontext() (pynlpl.formats.folia.New method), 979

leftcontext() (pynlpl.formats.folia.Note method), 259	leftcontext() (pynlpl.formats.folia.TextMarkupGap
leftcontext() (pynlpl.formats.folia.Observation method),	method), 811
568	leftcontext() (pynlpl.formats.folia.TextMarkupString
leftcontext() (pynlpl.formats.folia.ObservationLayer	method), 822
method), 698	leftcontext() (pynlpl.formats.folia.TextMarkupStyle
leftcontext() (pynlpl.formats.folia.Original method), 990	method), 833
leftcontext() (pynlpl.formats.folia.Paragraph method), 272	leftcontext() (pynlpl.formats.folia.TimeSegment method), 638
leftcontext() (pynlpl.formats.folia.Part method), 285	leftcontext() (pynlpl.formats.folia.TimingLayer method),
leftcontext() (pynlpl.formats.folia.PhonContent method),	756
509	leftcontext() (pynlpl.formats.folia.Whitespace method),
leftcontext() (pynlpl.formats.folia.PosAnnotation	404
method), 443	leftcontext() (pynlpl.formats.folia.Word method), 418
leftcontext() (pynlpl.formats.folia.Predicate method), 580	lemma() (pynlpl.formats.folia.Word method), 418
leftcontext() (pynlpl.formats.folia.Quote method), 298	LemmaAnnotation (class in pynlpl.formats.folia), 459
leftcontext() (pynlpl.formats.folia.Reference method),	levenshtein() (in module pynlpl.statistics), 1064
311	license() (pynlpl.formats.folia.Document method), 19
leftcontext() (pynlpl.formats.folia.Row method), 324	Linebreak (class in pynlpl.formats.folia), 212
leftcontext() (pynlpl.formats.folia.SemanticRole	List (class in pynlpl.formats.folia), 225
method), 626	ListItem (class in pynlpl.formats.folia), 238
leftcontext() (pynlpl.formats.folia.SemanticRolesLayer	LMClient (class in pynlpl.lm.client), 1058
method), 745	load() (pynlpl.formats.folia.Document method), 19
leftcontext() (pynlpl.formats.folia.SenseAnnotation method), 476	load() (pynlpl.lm.lm.SimpleLanguageModel method),
leftcontext() (pynlpl.formats.folia.Sentence method), 338	load() (pynlpl.statistics.FrequencyList method), 1063
leftcontext() (pynlpl.formats.folia.Sentiment method),	log() (in module pynlpl.common), 3
591	log2() (in module pynlpl.statistics), 1064
leftcontext() (pynlpl.formats.folia.SentimentLayer	logscore() (pynlpl.lm.srilm.SRILM method), 1057
method), 709	
leftcontext() (pynlpl.formats.folia.Statement method),	M
603	mae() (in module pynlpl.evaluation), 11
leftcontext() (pynlpl.formats.folia.StatementLayer	mae() (pynlpl.evaluation.OrdinalEvaluation method), 10
method), 721	MarkovChain (class in pynlpl.statistics), 1063
leftcontext() (pynlpl.formats.folia.SubjectivityAnnotation method), 487	maxentropy() (pynlpl.statistics.Distribution method), 1062
leftcontext()  (pynlpl.formats.folia. Suggestion  method),	mean() (in module pynlpl.statistics), 1064
1001	median() (in module pynlpl.statistics), 1064
leftcontext() (pynlpl.formats.folia.SynsetFeature method), 877	mergewords() (pynlpl.formats.folia.Sentence method), 338
leftcontext() (pynlpl.formats.folia.SyntacticUnit method),	Metric (class in pynlpl.formats.folia), 1040
615	mode() (in module pynlpl.statistics), 1064
leftcontext() (pynlpl.formats.folia.SyntaxLayer method),	mode() (pynlpl.statistics.Distribution method), 1062
733	mode() (pynlpl.statistics.FrequencyList method), 1063
leftcontext() (pynlpl.formats.folia.Table method), 352	morpheme() (pynlpl.formats.folia.Word method), 418
leftcontext() (pynlpl.formats.folia.TableHead method),	morphemes() (pynlpl.formats.folia.Word method), 418
378	MultiWindower (class in pynlpl.textprocessors), 1066
leftcontext() (pynlpl.formats.folia.Term method), 365	MultiWordAlignment (class in pynlpl.formats.giza), 1053
leftcontext() (pynlpl.formats.folia.Text method), 391 leftcontext() (pynlpl.formats.folia.TextContent method),	N
498	
leftcontext() (pynlpl.formats.folia.TextMarkupCorrection	New (class in pynlpl.formats.folia), 973 new() (pynlpl.formats.folia.Correction method), 947
method), 843	new () (pympinormaishona.concentin memod), 34/
16	next() (pynlpl.formats.folia AbstractAnnotationLaver
leftcontext() (pynlpl.formats.folia.TextMarkupError method), 854	next() (pynlpl.formats.folia.AbstractAnnotationLayer method), 79

next()	(pynlpl.formats.folia.AbstractSpanAnnotation	465
	method), 56	next() (pynlpl.formats.folia.Linebreak method), 220
next()	(pynlpl.formats.folia.AbstractStructureElement	next() (pynlpl.formats.folia.List method), 233
	method), 42	next() (pynlpl.formats.folia.ListItem method), 246
next()	(pynlpl.formats.folia.AbstractTextMarkup	next() (pynlpl.formats.folia.Metric method), 1046
	method), 89	next() (pynlpl.formats.folia.New method), 979
next()		next() (pynlpl.formats.folia.Note method), 259
()	method), 67	next() (pynlpl.formats.folia.Observation method), 568
next()	(pynlpl.formats.folia.ActorFeature method), 888	next() (pynlpl.formats.folia.ObservationLayer method),
	(pynlpl.formats.folia.Alignment method), 1012	698
next()		next() (pynlpl.formats.folia.Original method), 990
next()	1023	next() (pynlpl.formats.folia.Paragraph method), 272
nevt()	(pynlpl.formats.folia.Alternative method), 923	next() (pynlpl.formats.folia.Part method), 285
	(pynlpl.formats.folia.AlternativeLayers method),	next() (pynlpl.formats.folia.PhonContent method), 509
псхі()	935	next() (pynlpl.formats.folia.PosAnnotation method), 443
novt()		next() (pynlpl.formats.folia.Predicate method), 580
next()	1, 1	
	method), 899	next() (pynlpl.formats.folia.Quote method), 298
	(pynlpl.formats.folia.Cell method), 105	next() (pynlpl.formats.folia.Reference method), 311
	(pynlpl.formats.folia.Chunk method), 522	next() (pynlpl.formats.folia.Row method), 324
	(pynlpl.formats.folia.ChunkingLayer method), 650	next() (pynlpl.formats.folia.SemanticRole method), 626
next()	(pynlpl.formats.folia.CoreferenceChain method),	next() (pynlpl.formats.folia.SemanticRolesLayer
	533	method), 745
next()	1. 1	next() (pynlpl.formats.folia.SenseAnnotation method),
	662	476
next()	1. 1	next() (pynlpl.formats.folia.Sentence method), 339
	768	next() (pynlpl.formats.folia.Sentiment method), 591
	(pynlpl.formats.folia.Correction method), 947	next() (pynlpl.formats.folia.SentimentLayer method), 709
	(pynlpl.formats.folia.Current method), 957	next() (pynlpl.formats.folia.Statement method), 603
	(pynlpl.formats.folia.Definition method), 118	next() (pynlpl.formats.folia.StatementLayer method), 721
next()	(pynlpl.formats.folia.DependenciesLayer method),	next() (pynlpl.formats.folia.SubjectivityAnnotation
	674	method), 487
next()	(pynlpl.formats.folia.Dependency method), 545	next() (pynlpl.formats.folia.Suggestion method), 1001
next()	(pynlpl.formats.folia.DependencyDependent	next() (pynlpl.formats.folia.SynsetFeature method), 877
	method), 780	next() (pynlpl.formats.folia.SyntacticUnit method), 615
next()	(pynlpl.formats.folia.Description method), 1035	next() (pynlpl.formats.folia.SyntaxLayer method), 733
next()	(pynlpl.formats.folia.Division method), 131	next() (pynlpl.formats.folia.Table method), 352
next()	(pynlpl.formats.folia.DomainAnnotation method),	next() (pynlpl.formats.folia.TableHead method), 378
	432	next() (pynlpl.formats.folia.Term method), 365
next()	(pynlpl.formats.folia.EnddatetimeFeature method),	next() (pynlpl.formats.folia.Text method), 391
	910	next() (pynlpl.formats.folia.TextContent method), 498
next()	(pynlpl.formats.folia.EntitiesLayer method), 686	next() (pynlpl.formats.folia.TextMarkupCorrection
	(pynlpl.formats.folia.Entity method), 556	method), 844
	(pynlpl.formats.folia.Entry method), 144	next() (pynlpl.formats.folia.TextMarkupError method),
	(pynlpl.formats.folia.ErrorDetection method), 968	854
	(pynlpl.formats.folia.Event method), 157	next() (pynlpl.formats.folia.TextMarkupGap method),
	(pynlpl.formats.folia.Example method), 170	812
	(pynlpl.formats.folia.Feature method), 866	next() (pynlpl.formats.folia.TextMarkupString method),
	(pynlpl.formats.folia.Figure method), 183	822
	(pynlpl.formats.folia.Gap method), 194	next() (pynlpl.formats.folia.TextMarkupStyle method),
4,5	(pynlpl.formats.folia.Head method), 207	833
	(pynlpl.formats.folia.Headspan method), 791	next() (pynlpl.formats.folia.TimeSegment method), 638
next()		next() (pynlpl.formats.folia.TimingLayer method), 756
полц	454	next() (pynlpl.formats.folia.Whitespace method), 404
next()	(pynlpl.formats.folia.LemmaAnnotation method),	next() (pynlpl.formats.folia.Word method), 419
110/11/1	(P, inpritoring to inches in in a finite in the interior).	iiona, , aprilipinionium. nona monana, 117

next() (pynlpl.formats.taggerdata.Taggerdata method),	OCCURRENCES (pynlpl.formats.folia.Dependency attribute), 541
normalize() (in module pynlpl.statistics), 1064	$OCCURRENCES\ (pynlpl.formats.folia. Dependency Dependent$
Note (class in pynlpl.formats.folia), 251	attribute), 776
ns() (in module pynlpl.formats.sonar), 1055	OCCURRENCES (pynlpl.formats.folia.Description at-
	tribute), 1031
0	OCCURRENCES (pynlpl.formats.folia.Division at-
Observation (class in pynlpl.formats.folia), 562	tribute), 126
ObservationLayer (class in pynlpl.formats.folia), 691	OCCURRENCES (pynlpl.formats.folia.DomainAnnotation
OCCURRENCES (pynlpl.formats.folia.AbstractAnnotation	Laver attribute), 428
attribute), 74	OCCURRENCES (pynlpl.formats.folia.EnddatetimeFeature
OCCURRENCES (pynlpl.formats.folia.AbstractElement	attribute), 907
attribute), 26	OCCURRENCES (pynlpl.formats.folia.EntitiesLayer at-
OCCURRENCES (pynlpl.formats.folia.AbstractSpanAnnot	
attribute), 52	OCCURRENCES (pynlpl.formats.folia.Entity attribute),
OCCURRENCES (pynlpl.formats.folia.AbstractStructureE	
	OCCURRENCES (pynlpl.formats.folia.Entry attribute),
attribute), 37	100
OCCURRENCES (pynlpl.formats.folia.AbstractTextMarku	OCCURRENCES (pynlpl.formats.folia.ErrorDetection
attribute), 86	
OCCURRENCES (pynlpl.formats.folia.AbstractTokenAnno	OCCURRENCES (pynlpl.formats.folia.Event attribute),
attribute), 63	152
OCCURRENCES (pynlpl.formats.folia.ActorFeature at-	
tribute), 885	OCCURRENCES (pynlpl.formats.folia.Example at-
OCCURRENCES (pynlpl.formats.folia.Alignment	tribute), 165
attribute), 1009	OCCURRENCES (pynlpl.formats.folia.Feature at-
OCCURRENCES (pynlpl.formats.folia.AlignReference	tribute), 862
attribute), 1020	OCCURRENCES (pynlpl.formats.folia.Figure attribute),
OCCURRENCES (pynlpl.formats.folia.Alternative at-	178
tribute), 919	OCCURRENCES (pynlpl.formats.folia.Gap attribute),
OCCURRENCES (pynlpl.formats.folia.AlternativeLayers	191
attribute), 931	OCCURRENCES (pynlpl.formats.folia.Head attribute),
OCCURRENCES (pynlpl.formats.folia.BegindatetimeFeatu	re 202
attribute), 896	OCCURRENCES (pynlpl.formats.folia.Headspan at-
OCCURRENCES (pynlpl.formats.folia.Cell attribute),	tribute), 788
100	OCCURRENCES (pynlpl.formats.folia.LangAnnotation
OCCURRENCES (pynlpl.formats.folia.Chunk attribute),	attribute), 450
518	OCCURRENCES (pynlpl.formats.folia.LemmaAnnotation
OCCURRENCES (pynlpl.formats.folia.ChunkingLayer	attribute), 461
attribute), 646	OCCURRENCES (pynlpl.formats.folia.Linebreak
OCCURRENCES (pynlpl.formats.folia.CoreferenceChain	attribute), 215
attribute), 529	OCCURRENCES (pynlpl.formats.folia.List attribute),
OCCURRENCES (pynlpl.formats.folia.CoreferenceLayer	228
attribute), 658	OCCURRENCES (pynlpl.formats.folia.ListItem at-
OCCURRENCES (pynlpl.formats.folia.CoreferenceLink	tribute), 241
attribute), 764	OCCURRENCES (pynlpl.formats.folia.Metric attribute),
	1042
4.0 1	OCCURRENCES (pynlpl.formats.folia.New attribute),
attribute), 944	975
OCCURRENCES (pynlpl.formats.folia.Current at-	OCCURRENCES (pynlpl.formats.folia.Note attribute),
tribute), 953	254
OCCURRENCES (pynlpl.formats.folia.Definition	
attribute), 113	OCCURRENCES (pynlpl.formats.folia.Observation attribute), 564
OCCURRENCES (pynlpl.formats.folia.DependenciesLayer	OCCURRENCES (pynlpl.formats.folia.ObservationLayer
attribute), 670	** *
	attribute), 693

OCCURRENCES (pynlpl.formats.folia.Original attribute), 986	OCCURRENCES (pynlpl.formats.folia.TextMarkupCorrection attribute), 840
OCCURRENCES (pynlpl.formats.folia.Paragraph	OCCURRENCES (pynlpl.formats.folia.TextMarkupError
attribute), 267	attribute), 851
OCCURRENCES (pynlpl.formats.folia.Part attribute),	OCCURRENCES (pynlpl.formats.folia.TextMarkupGap
280	attribute), 808
OCCURRENCES (pynlpl.formats.folia.PhonContent at-	OCCURRENCES (pynlpl.formats.folia.TextMarkupString
tribute), 505	attribute), 819
OCCURRENCES (pynlpl.formats.folia.PosAnnotation	OCCURRENCES (pynlpl.formats.folia.TextMarkupStyle
attribute), 439	attribute), 830
OCCURRENCES (pynlpl.formats.folia.Predicate at-	OCCURRENCES (pynlpl.formats.folia.TimeSegment at-
tribute), 576	tribute), 634
OCCURRENCES (pynlpl.formats.folia.Quote attribute),	OCCURRENCES (pynlpl.formats.folia.TimingLayer at-
293	tribute), 752
OCCURRENCES (pynlpl.formats.folia.Reference	OCCURRENCES (pynlpl.formats.folia.Whitespace at-
attribute), 306	tribute), 399
OCCURRENCES (pynlpl.formats.folia.Row attribute),	OCCURRENCES (pynlpl.formats.folia.Word attribute),
319	413
OCCURRENCES (pynlpl.formats.folia.SemanticRole at-	OCCURRENCES_PER_SET
tribute), 623	(pynlpl.formats.folia.AbstractAnnotationLayer
OCCURRENCES (pynlpl.formats.folia.SemanticRolesLayo	
attribute), 740	OCCURRENCES_PER_SET
OCCURRENCES (pynlpl.formats.folia.SenseAnnotation	(pynlpl.formats.folia.AbstractElement at-
attribute), 472	tribute), 26
OCCURRENCES (pynlpl.formats.folia.Sentence at-	OCCURRENCES_PER_SET
tribute), 332	(pynlpl.formats.folia.AbstractSpanAnnotation
OCCURRENCES (pynlpl.formats.folia.Sentiment	attribute), 52
attribute), 588	OCCURRENCES_PER_SET
OCCURRENCES (pynlpl.formats.folia.SentimentLayer	(pynlpl.formats.folia.AbstractStructureElement
attribute), 705 OCCURRENCES (pynlpl.formats.folia.Statement at-	attribute), 37 OCCURRENCES_PER_SET
OCCURRENCES (pynlpl.formats.folia.Statement attribute), 599	(pynlpl.formats.folia.AbstractTextMarkup
OCCURRENCES (pynlpl.formats.folia.StatementLayer	attribute), 86
attribute), 717	OCCURRENCES_PER_SET
OCCURRENCES (pynlpl.formats.folia.SubjectivityAnnota	
attribute), 483	attribute), 63
OCCURRENCES (pynlpl.formats.folia.Suggestion at-	
tribute), 997	(pynlpl.formats.folia.ActorFeature attribute),
OCCURRENCES (pynlpl.formats.folia.SynsetFeature at-	885
tribute), 873	OCCURRENCES_PER_SET
OCCURRENCES (pynlpl.formats.folia.SyntacticUnit at-	(pynlpl.formats.folia.Alignment attribute),
tribute), 611	1009
OCCURRENCES (pynlpl.formats.folia.SyntaxLayer at-	OCCURRENCES_PER_SET
tribute), 729	(pynlpl.formats.folia.AlignReference attribute),
OCCURRENCES (pynlpl.formats.folia.Table attribute),	1020
347	OCCURRENCES_PER_SET
OCCURRENCES (pynlpl.formats.folia.TableHead	(pynlpl.formats.folia.Alternative attribute),
attribute), 373	919
OCCURRENCES (pynlpl.formats.folia.Term attribute),	OCCURRENCES_PER_SET
360	(pynlpl.formats.folia.AlternativeLayers at-
OCCURRENCES (pynlpl.formats.folia.Text attribute),	tribute), 931
386 OCCUPPENCES (number formate folio ToytContent, et	OCCURRENCES_PER_SET
OCCURRENCES (pynlpl.formats.folia.TextContent attribute), 495	(pynlpl.formats.folia.BegindatetimeFeature attribute), 896
いいしにょ サブン	atti 10ttc), 070

OCCURRENCES_PER_SET (pynlpl.formats.folia.Cell	964
attribute), 100	OCCURRENCES_PER_SET (pynlpl.formats.folia.Event
OCCURRENCES_PER_SET	attribute), 152
(pynlpl.formats.folia.Chunk attribute), 518	OCCURRENCES_PER_SET
OCCURRENCES_PER_SET	(pynlpl.formats.folia.Example attribute),
(pynlpl.formats.folia.ChunkingLayer attribute),	165
646	OCCURRENCES_PER_SET
OCCURRENCES_PER_SET	(pynlpl.formats.folia.Feature attribute), 862 OCCURRENCES PER SET
(pynlpl.formats.folia.CoreferenceChain attribute), 529	(pynlpl.formats.folia.Figure attribute), 178
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET (pynlpl.formats.folia.Gap
(pynlpl.formats.folia.CoreferenceLayer at-	attribute), 191
tribute), 658	OCCURRENCES_PER_SET (pynlpl.formats.folia.Head
OCCURRENCES_PER_SET	attribute), 202
(pynlpl.formats.folia.CoreferenceLink at-	OCCURRENCES_PER_SET
tribute), 764	(pynlpl.formats.folia.Headspan attribute),
OCCURRENCES_PER_SET	788
(pynlpl.formats.folia.Correction attribute),	OCCURRENCES_PER_SET
944	(pynlpl.formats.folia.LangAnnotation at-
OCCURRENCES_PER_SET	tribute), 450
(pynlpl.formats.folia.Current attribute), 953	OCCURRENCES_PER_SET
OCCURRENCES_PER_SET	(pynlpl.formats.folia.LemmaAnnotation at-
(pynlpl.formats.folia.Definition attribute),	tribute), 461 OCCURRENCES_PER_SET
OCCURRENCES_PER_SET	(pynlpl.formats.folia.Linebreak attribute),
(pynlpl.formats.folia.DependenciesLayer	215
attribute), 670	OCCURRENCES_PER_SET (pynlpl.formats.folia.List
OCCURRENCES_PER_SET	attribute), 228
(pynlpl.formats.folia.Dependency attribute),	OCCURRENCES_PER_SET
541	(pynlpl.formats.folia.ListItem attribute),
OCCURRENCES_PER_SET	241
(pynlpl.formats.folia.DependencyDependent	OCCURRENCES_PER_SET
attribute), 776	(pynlpl.formats.folia.Metric attribute), 1042
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET (pynlpl.formats.folia.New
(pynlpl.formats.folia.Description attribute), 1031	attribute), 975 OCCURRENCES_PER_SET (pynlpl.formats.folia.Note
OCCURRENCES_PER_SET	attribute), 254
	OCCURRENCES_PER_SET
126	(pynlpl.formats.folia.Observation attribute),
OCCURRENCES_PER_SET	564
(pynlpl.formats.folia.DomainAnnotation	OCCURRENCES_PER_SET
attribute), 428	(pynlpl.formats.folia.ObservationLayer at-
OCCURRENCES_PER_SET	tribute), 694
(pynlpl.formats.folia.EnddatetimeFeature	OCCURRENCES_PER_SET
attribute), 907	(pynlpl.formats.folia.Original attribute),
OCCURRENCES_PER_SET	986 OCCURRENCES DED SET
(pynlpl.formats.folia.EntitiesLayer attribute), 682	OCCURRENCES_PER_SET (pynlpl.formats.folia.Paragraph attribute),
OCCURRENCES_PER_SET	(pynipi.ioiniats.ioiia.raragraph attribute),
(pynlpl.formats.folia.Entity attribute), 553	OCCURRENCES_PER_SET (pynlpl.formats.folia.Part
OCCURRENCES_PER_SET (pynlpl.formats.folia.Entry	attribute), 280
attribute), 139	OCCURRENCES_PER_SET
OCCURRENCES_PER_SET	(pynlpl.formats.folia.PhonContent attribute),
(pynlpl.formats.folia.ErrorDetection attribute),	505

OCCURRENCES_PER_SET	OCCURRENCES_PER_SET
(pynlpl.formats.folia.PosAnnotation attribute), 439	(pynlpl.formats.folia.TableHead attribute), 373
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET (pynlpl.formats.folia.Term
(pynlpl.formats.folia.Predicate attribute),	attribute), 360
576	OCCURRENCES_PER_SET (pynlpl.formats.folia.Text
OCCURRENCES_PER_SET	attribute), 386
(pynlpl.formats.folia.Quote attribute), 293	OCCURRENCES_PER_SET
OCCURRENCES_PER_SET	(pynlpl.formats.folia.TextContent attribute),
(pynlpl.formats.folia.Reference attribute), 306	495 OCCUPRENCES DED SET
OCCURRENCES_PER_SET (pynlpl.formats.folia.Row	OCCURRENCES_PER_SET (pynlpl.formats.folia.TextMarkupCorrection
attribute), 319	attribute), 840
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET
(pynlpl.formats.folia.SemanticRole attribute),	(pynlpl.formats.folia.TextMarkupError at-
623	tribute), 851
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET
(pynlpl.formats.folia.SemanticRolesLayer	(pynlpl.formats.folia.TextMarkupGap at-
attribute), 741	tribute), 808
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET
(pynlpl.formats.folia.SenseAnnotation at-	(pynlpl.formats.folia.TextMarkupString at-
tribute), 472 OCCURRENCES_PER_SET	tribute), 819
(pynlpl.formats.folia.Sentence attribute),	OCCURRENCES_PER_SET (pynlpl.formats.folia.TextMarkupStyle at-
332	tribute), 830
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET
(pynlpl.formats.folia.Sentiment attribute), 588	(pynlpl.formats.folia.TimeSegment attribute), 634
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET
(pynlpl.formats.folia.SentimentLayer attribute), 705	(pynlpl.formats.folia.TimingLayer attribute), 752
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET
(pynlpl.formats.folia.Statement attribute), 599	(pynlpl.formats.folia.Whitespace attribute), 399
OCCURRENCES_PER_SET	OCCURRENCES_PER_SET (pynlpl.formats.folia.Word
(pynlpl.formats.folia.StatementLayer attribute),	attribute), 413
717	$OPTIONAL\_ATTRIBS\ (pynlpl.formats.folia. AbstractAnnotation Layer$
OCCURRENCES_PER_SET	attribute), 74
(pynlpl.formats.folia.SubjectivityAnnotation	OPTIONAL_ATTRIBS (pynlpl.formats.folia.AbstractElement
attribute), 483	attribute), 26
OCCURRENCES_PER_SET (pynlpl.formats.folia.Suggestion attribute),	OPTIONAL_ATTRIBS (pynlpl.formats.folia.AbstractSpanAnnotation attribute), 52
(pyinpi.iormats.iona.suggestion attribute),	OPTIONAL_ATTRIBS (pynlpl.formats.folia.AbstractStructureElement
OCCURRENCES_PER_SET	attribute), 37
(pynlpl.formats.folia.SynsetFeature attribute),	OPTIONAL_ATTRIBS (pynlpl.formats.folia.AbstractTextMarkup
873	attribute), 86
OCCURRENCES_PER_SET	$OPTIONAL\_ATTRIBS\ (pynlpl. formats. folia. Abstract Token Annotation$
(pynlpl.formats.folia.SyntacticUnit attribute),	attribute), 63
611	OPTIONAL_ATTRIBS (pynlpl.formats.folia.ActorFeature
OCCURRENCES_PER_SET	attribute), 885
(pynlpl.formats.folia.SyntaxLayer attribute), 729	OPTIONAL_ATTRIBS (pynlpl.formats.folia.Alignment
OCCURRENCES_PER_SET (pynlpl.formats.folia.Table	attribute), 1009 OPTIONAL_ATTRIBS (pynlpl.formats.folia.AlignReference
attribute), 347	attribute), 1020

OPTIONAL ATTRIBS (pynlpl.formats.folia.Alternative OPTIONAL ATTRIBS (pynlpl.formats.folia.Gap attribute), 919 tribute), 191 OPTIONAL ATTRIBS (pynlpl.formats.folia.AlternativeLayOFTIONAL ATTRIBS (pynlpl.formats.folia.Head atattribute), 931 tribute), 202 OPTIONAL ATTRIBS (pynlpl.formats.folia.Begindatetime OPATIONAL ATTRIBS (pynlpl.formats.folia.Headspan attribute), 896 attribute), 788 OPTIONAL ATTRIBS (pynlpl.formats.folia.Cell at-OPTIONAL ATTRIBS (pynlpl.formats.folia.LangAnnotation tribute), 100 attribute), 450 OPTIONAL ATTRIBS (pynlpl.formats.folia.Chunk at- OPTIONAL ATTRIBS (pynlpl.formats.folia.LemmaAnnotation tribute), 518 attribute), 461 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.ChunkingLaye@PTIONAL\_ATTRIBS (pynlpl.formats.folia.Linebreak attribute), 215 attribute), 646 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.CoreferenceClaPfTIONAL\_ATTRIBS (pynlpl.formats.folia.List attribute), 529 tribute), 228 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.CoreferenceLaQPTIONAL\_ATTRIBS (pynlpl.formats.folia.ListItem atattribute), 658 tribute), 241 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.CoreferenceLi@PTIONAL\_ATTRIBS (pynlpl.formats.folia.Metric atattribute), 764 tribute), 1042 OPTIONAL ATTRIBS (pynlpl.formats.folia.Correction OPTIONAL ATTRIBS (pynlpl.formats.folia.New attribute), 944 attribute), 975 OPTIONAL ATTRIBS (pynlpl.formats.folia.Current at-OPTIONAL ATTRIBS (pynlpl.formats.folia.Note tribute), 953 attribute), 254 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Definition OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Observation attribute), 113 attribute), 564 OPTIONAL ATTRIBS (pynlpl.formats.folia.Dependencies DPFFONAL ATTRIBS (pynlpl.formats.folia.ObservationLayer attribute), 670 attribute), 694 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Dependency OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Original atattribute), 541 tribute), 986 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.DependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDepende (pynlpl.formats.folia.Paragraph attribute), 776 attribute), 267 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Description OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Part atattribute), 1031 tribute), 280 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Division at-OPTIONAL\_ATTRIBS (pynlpl.formats.folia.PhonContent tribute), 126 attribute), 506 OPTIONAL ATTRIBS (pynlpl.formats.folia.DomainAnnot@@filONAL ATTRIBS (pynlpl.formats.folia.PosAnnotation attribute), 428 attribute), 439 OPTIONAL ATTRIBS (pynlpl.formats.folia.EnddatetimeF@RFFONAL ATTRIBS (pynlpl.formats.folia.Predicate attribute), 907 attribute), 576 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.EntitiesLayer OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Quote atattribute), 682 tribute), 293 OPTIONAL ATTRIBS (pynlpl.formats.folia.Entity at-OPTIONAL ATTRIBS (pynlpl.formats.folia.Reference tribute), 553 attribute), 306 OPTIONAL ATTRIBS (pynlpl.formats.folia.Entry at-OPTIONAL ATTRIBS (pynlpl.formats.folia.Row tribute), 139 attribute), 319 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.ErrorDetectiorOPTIONAL\_ATTRIBS (pynlpl.formats.folia.SemanticRole attribute), 964 attribute), 623 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Event at-OPTIONAL\_ATTRIBS (pynlpl.formats.folia.SemanticRolesLayer tribute), 152 attribute), 741 (pynlpl.formats.folia.Example OPTIONAL\_ATTRIBS (pynlpl.formats.folia.SenseAnnotation OPTIONAL\_ATTRIBS attribute), 165 attribute), 472 OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Feature at-OPTIONAL\_ATTRIBS (pynlpl.formats.folia.Sentence tribute), 862 attribute), 332 OPTIONAL ATTRIBS (pynlpl.formats.folia.Figure at-OPTIONAL ATTRIBS (pynlpl.formats.folia.Sentiment tribute), 178 attribute), 588

OPTIONAL_ATTRIBS (pynlpl.formats.folia.SentimentLayer method), 42
attribute), 705 originaltext() (pynlpl.formats.folia.AbstractTextMarkup
OPTIONAL_ATTRIBS (pynlpl.formats.folia.Statement method), 90
attribute), 599 originaltext() (pynlpl.formats.folia.AbstractTokenAnnotation
OPTIONAL_ATTRIBS (pynlpl.formats.folia.StatementLayer method), 67
attribute), 717 originaltext() (pynlpl.formats.folia.ActorFeature
OPTIONAL_ATTRIBS (pynlpl.formats.folia.SubjectivityAnnotation method), 888
attribute), 483 originaltext() (pynlpl.formats.folia.Alignment method),
OPTIONAL_ATTRIBS (pynlpl.formats.folia.Suggestion 1012
attribute), 997 originaltext() (pynlpl.formats.folia.AlignReference
OPTIONAL_ATTRIBS (pynlpl.formats.folia.SynsetFeature method), 1024
attribute), 873 originaltext() (pynlpl.formats.folia.Alternative method),
OPTIONAL_ATTRIBS (pynlpl.formats.folia.SyntacticUnit 924
attribute), 611 originaltext() (pynlpl.formats.folia.AlternativeLayers
OPTIONAL_ATTRIBS (pynlpl.formats.folia.SyntaxLayer method), 935
attribute), 729 originaltext() (pynlpl.formats.folia.BegindatetimeFeature OPTIONAL_ATTRIBS (pynlpl.formats.folia.Table at method), 899
OPTIONAL_ATTRIBS (pynlpl.formats.folia.Table attribute), 347 method), 899  tribute), 347 originaltext() (pynlpl.formats.folia.Cell method), 105
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TableHead originaltext() (pynlpl.formats.folia.Cen method), 522
attribute), 373 originaltext() (pympi.formats.folia.ChunkingLayer
OPTIONAL_ATTRIBS (pynlpl.formats.folia.Term at- method), 651
tribute), 360 originaltext() (pynlpl.formats.folia.CoreferenceChain
OPTIONAL_ATTRIBS (pynlpl.formats.folia.Text at- method), 533
tribute), 386 originaltext() (pynlpl.formats.folia.CoreferenceLayer
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TextContent method), 662
attribute), 495 originaltext() (pynlpl.formats.folia.CoreferenceLink
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TextMarkupCorrection method), 768
attribute), 840 originaltext() (pynlpl.formats.folia.Correction method),
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TextMarkupError 948
attribute), 851 originaltext() (pynlpl.formats.folia.Current method), 957
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TextMarkupGappiginaltext() (pynlpl.formats.folia.Definition method),
attribute), 808
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TextMarkupStririginaltext() (pynlpl.formats.folia.DependenciesLayer
attribute), 819 method), 674
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TextMarkupStwieginaltext() (pynlpl.formats.folia.Dependency method), attribute), 830
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TimeSegment originaltext() (pynlpl.formats.folia.DependencyDependent attribute), 634 method), 780
OPTIONAL_ATTRIBS (pynlpl.formats.folia.TimingLayer originaltext() (pynlpl.formats.folia.Description method),
attribute), 752
OPTIONAL_ATTRIBS (pynlpl.formats.folia.Whitespace originaltext() (pynlpl.formats.folia.Division method), 131
attribute), 399 originaltext() (pynlpl.formats.folia.DomainAnnotation
OPTIONAL_ATTRIBS (pynlpl.formats.folia.Word atmethod), 432
tribute), 413 originaltext() (pynlpl.formats.folia.EnddatetimeFeature
OrdinalEvaluation (class in pynlpl.evaluation), 10 method), 910
Original (class in pynlpl.formats.folia), 984 originaltext() (pynlpl.formats.folia.EntitiesLayer
original() (pynlpl.formats.folia.Correction method), 947 method), 686
originaltext() (pynlpl.formats.folia.AbstractAnnotationLayeroriginaltext() (pynlpl.formats.folia.Entity method), 557
method), 79 originaltext() (pynlpl.formats.folia.Entry method), 144
originaltext() (pynlpl.formats.folia.AbstractElement originaltext() (pynlpl.formats.folia.ErrorDetection
method), 29 method), 968
originaltext() (pynlpl.formats.folia.AbstractSpanAnnotation originaltext() (pynlpl.formats.folia.Event method), 157
method), 56 originaltext() (pynlpl.formats.folia.Example method),
originaltext() (pynlpl.formats.folia.AbstractStructureElement 170

originaltext() (pynlpl.formats.folia.Feature method), 866 originaltext() (pynlpl.formats.folia.Figure method), 183 originaltext() (pynlpl.formats.folia.Gap method), 194	originaltext() (pynlpl.formats.folia.SynsetFeature method), 877
originaltext() (pynlpl.formats.folia.Head method), 207 originaltext() (pynlpl.formats.folia.Headspan method),	originaltext() (pynlpl.formats.folia.SyntacticUnit method), 615
originaltext() (pynlpl.formats.folia.LangAnnotation	originaltext() (pynlpl.formats.folia.SyntaxLayer method), 733
method), 454 originaltext() (pynlpl.formats.folia.LemmaAnnotation method), 465	originaltext() (pynlpl.formats.folia.Table method), 352 originaltext() (pynlpl.formats.folia.TableHead method), 378
originaltext() (pynlpl.formats.folia.Linebreak method),	originaltext() (pynlpl.formats.folia.Term method), 365 originaltext() (pynlpl.formats.folia.Text method), 391
originaltext() (pynlpl.formats.folia.List method), 233 originaltext() (pynlpl.formats.folia.ListItem method), 246	originaltext() (pynlpl.formats.folia.TextContent method), 499
originaltext() (pynlpl.formats.folia.Metric method), 1046 originaltext() (pynlpl.formats.folia.New method), 979	originaltext() (pynlpl.formats.folia.TextMarkupCorrection method), 844
originaltext() (pynlpl.formats.folia.Note method), 259 originaltext() (pynlpl.formats.folia.Observation method),	originaltext() (pynlpl.formats.folia.TextMarkupError method), 854
originaltext() (pynlpl.formats.folia.ObservationLayer	originaltext() (pynlpl.formats.folia.TextMarkupGap method), 812
method), 698 originaltext() (pynlpl.formats.folia.Original method), 990	originaltext() (pynlpl.formats.folia.TextMarkupString method), 822
originaltext() (pynlpl.formats.folia.Paragraph method),	originaltext() (pynlpl.formats.folia.TextMarkupStyle method), 833
originaltext() (pynlpl.formats.folia.Part method), 285 originaltext() (pynlpl.formats.folia.PhonContent method),	originaltext() (pynlpl.formats.folia.TimeSegment method), 638
originaltext() (pynlpl.formats.folia.PosAnnotation	originaltext() (pynlpl.formats.folia.TimingLayer method), 757
method), 443	originaltext() (pynlpl.formats.folia.Whitespace method),
originaltext() (pynlpl.formats.folia.Predicate method),	404
580 originaltext() (pynlpl.formats.folia.Quote method), 298	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062
580 originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation
580 originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method),	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745 originaltext() (pynlpl.formats.folia.SenseAnnotation method), 476	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063 Paragraph (class in pynlpl.formats.folia), 264 paragraph() (pynlpl.formats.folia.Sentence method), 339
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745 originaltext() (pynlpl.formats.folia.SenseAnnotation method), 476 originaltext() (pynlpl.formats.folia.Sentence method), 339	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063 Paragraph (class in pynlpl.formats.folia), 264
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745 originaltext() (pynlpl.formats.folia.SenseAnnotation method), 476 originaltext() (pynlpl.formats.folia.Sentence method), 339 originaltext() (pynlpl.formats.folia.Sentiment method), 592	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063 Paragraph (class in pynlpl.formats.folia), 264 paragraph() (pynlpl.formats.folia.Sentence method), 339 paragraph() (pynlpl.formats.folia.Word method), 419
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745 originaltext() (pynlpl.formats.folia.SenseAnnotation method), 476 originaltext() (pynlpl.formats.folia.Sentence method), 339 originaltext() (pynlpl.formats.folia.Sentiment method), 592 originaltext() (pynlpl.formats.folia.Sentiment Layer method), 710	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063 Paragraph (class in pynlpl.formats.folia), 264 paragraph() (pynlpl.formats.folia.Sentence method), 339 paragraph() (pynlpl.formats.folia.Word method), 419 paragraphs() (pynlpl.formats.folia.AbstractStructureElement method), 42
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745 originaltext() (pynlpl.formats.folia.SenseAnnotation method), 476 originaltext() (pynlpl.formats.folia.Sentence method), 339 originaltext() (pynlpl.formats.folia.Sentiment method), 592 originaltext() (pynlpl.formats.folia.Sentiment method), 710 originaltext() (pynlpl.formats.folia.Statement method), 603	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063 Paragraph (class in pynlpl.formats.folia), 264 paragraph() (pynlpl.formats.folia.Sentence method), 339 paragraph() (pynlpl.formats.folia.Word method), 419 paragraphs() (pynlpl.formats.folia.AbstractStructureElement method), 42 paragraphs() (pynlpl.formats.folia.Cell method), 105 paragraphs() (pynlpl.formats.folia.Definition method), 118 paragraphs() (pynlpl.formats.folia.Division method), 131 paragraphs() (pynlpl.formats.folia.Document method), 19
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745 originaltext() (pynlpl.formats.folia.SenseAnnotation method), 476 originaltext() (pynlpl.formats.folia.Sentence method), 339 originaltext() (pynlpl.formats.folia.Sentiment method), 592 originaltext() (pynlpl.formats.folia.SentimentLayer method), 710 originaltext() (pynlpl.formats.folia.Statement method), 603 originaltext() (pynlpl.formats.folia.StatementLayer method), 721	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063 Paragraph (class in pynlpl.formats.folia), 264 paragraph() (pynlpl.formats.folia.Sentence method), 339 paragraph() (pynlpl.formats.folia.Word method), 419 paragraphs() (pynlpl.formats.folia.AbstractStructureElement method), 42 paragraphs() (pynlpl.formats.folia.Cell method), 105 paragraphs() (pynlpl.formats.folia.Definition method), 118 paragraphs() (pynlpl.formats.folia.Division method), 131
originaltext() (pynlpl.formats.folia.Quote method), 298 originaltext() (pynlpl.formats.folia.Reference method), 311 originaltext() (pynlpl.formats.folia.Row method), 324 originaltext() (pynlpl.formats.folia.SemanticRole method), 627 originaltext() (pynlpl.formats.folia.SemanticRolesLayer method), 745 originaltext() (pynlpl.formats.folia.SenseAnnotation method), 476 originaltext() (pynlpl.formats.folia.Sentence method), 339 originaltext() (pynlpl.formats.folia.Sentiment method), 592 originaltext() (pynlpl.formats.folia.SentimentLayer method), 710 originaltext() (pynlpl.formats.folia.Statement method), 603 originaltext() (pynlpl.formats.folia.StatementLayer method), 603 originaltext() (pynlpl.formats.folia.StatementLayer	originaltext() (pynlpl.formats.folia.Word method), 419 output() (pynlpl.statistics.Distribution method), 1062 output() (pynlpl.statistics.FrequencyList method), 1063 outputmetrics() (pynlpl.evaluation.ClassEvaluation method), 10  P p() (pynlpl.statistics.FrequencyList method), 1063 p() (pynlpl.statistics.MarkovChain method), 1063 Paragraph (class in pynlpl.formats.folia), 264 paragraph() (pynlpl.formats.folia.Sentence method), 339 paragraph() (pynlpl.formats.folia.Word method), 419 paragraphs() (pynlpl.formats.folia.AbstractStructureElement method), 42 paragraphs() (pynlpl.formats.folia.Cell method), 105 paragraphs() (pynlpl.formats.folia.Definition method), 118 paragraphs() (pynlpl.formats.folia.Division method), 131 paragraphs() (pynlpl.formats.folia.Document method), 19 paragraphs() (pynlpl.formats.folia.Entry method), 144

- paragraphs() (pynlpl.formats.folia.Linebreak method), 220 paragraphs() (pynlpl.formats.folia.List method), 233 paragraphs() (pynlpl.formats.folia.ListItem method), 246 paragraphs() (pynlpl.formats.folia.Note method), 259 paragraphs() (pynlpl.formats.folia.Paragraph method), paragraphs() (pynlpl.formats.folia.Part method), 285 paragraphs() (pynlpl.formats.folia.Quote method), 298 paragraphs() (pynlpl.formats.folia.Reference method), paragraphs() (pynlpl.formats.folia.Row method), 324 paragraphs() (pynlpl.formats.folia.Sentence method), 339 paragraphs() (pynlpl.formats.folia.Table method), 352 paragraphs() (pynlpl.formats.folia.TableHead method), 378 paragraphs() (pynlpl.formats.folia.Term method), 365 paragraphs() (pynlpl.formats.folia.Text method), 391 paragraphs() (pynlpl.formats.folia.Whitespace method), paragraphs() (pynlpl.formats.folia.Word method), 419 paragraphs() (pynlpl.formats.sonar.CorpusDocument method), 1054 (pynlpl.formats.sonar.CorpusDocumentX paragraphs() method), 1054 ParamSearch (class in pynlpl.evaluation), 10 parse() (pynlpl.formats.fql.Query method), 803 parse\_cgn\_postag() (in module pynlpl.formats.cgn), 1053 parseAlignment() (in module pynlpl.formats.giza), 1054 parseDistribution() (pynlpl.formats.timbl.TimblOutput method), 1055 parsemetadata() (pynlpl.formats.folia.Document method), 20 parsesubmetadata() (pynlpl.formats.folia.Document method), 20 parsexml() (pynlpl.formats.folia.AbstractAnnotationLayer class method), 79 parsexml() (pynlpl.formats.folia.AbstractElement class method), 29 parsexml() (pynlpl.formats.folia.AbstractSpanAnnotation class method), 56 class method), 42 (pynlpl.formats.folia.AbstractTextMarkup parsexml() class method), 90 class method), 67 (pynlpl.formats.folia.ActorFeature parsexml() class method), 889 (pynlpl.formats.folia.Alignment parsexml() class method), 1012 parsexml() (pynlpl.formats.folia.AlignReference class method), 1024 (pynlpl.formats.folia.Alternative parsexml() class
- method), 924 parsexml() (pynlpl.formats.folia.AlternativeLayers class method), 935 (pynlpl.formats.folia.BegindatetimeFeature parsexml() class method), 900 parsexml() (pynlpl.formats.folia.Cell class method), 105 parsexml() (pynlpl.formats.folia.Chunk class method), parsexml() (pynlpl.formats.folia.ChunkingLayer class method), 651 parsexml() (pynlpl.formats.folia.CoreferenceChain class method), 533 parsexml() (pynlpl.formats.folia.CoreferenceLayer class method), 663 parsexml() (pynlpl.formats.folia.CoreferenceLink class method), 768 (pynlpl.formats.folia.Correction class parsexml() method), 948 parsexml() (pynlpl.formats.folia.Current class method), parsexml() (pynlpl.formats.folia.Definition class method), (pynlpl.formats.folia.DependenciesLayer parsexml() class method), 674 (pynlpl.formats.folia.Dependency parsexml() class method), 545 parsexml() (pynlpl.formats.folia.DependencyDependent class method), 780 (pynlpl.formats.folia.Description parsexml() class method), 1035 parsexml() (pynlpl.formats.folia.Division class method), 131 parsexml() (pynlpl.formats.folia.Document method), 20 parsexml() (pynlpl.formats.folia.DomainAnnotation class method), 432 (pynlpl.formats.folia.EnddatetimeFeature parsexml() class method), 911 parsexml() (pynlpl.formats.folia.EntitiesLayer method), 686 parsexml() (pynlpl.formats.folia.Entity class method), parsexml() (pynlpl.formats.folia.AbstractStructureElement parsexml() (pynlpl.formats.folia.Entry class method), 144 parsexml() (pynlpl.formats.folia.ErrorDetection class method), 968 parsexml() (pynlpl.formats.folia.Event class method), 157 parsexml() (pynlpl.formats.folia.AbstractTokenAnnotation parsexml() (pynlpl.formats.folia.Example class method), parsexml() (pynlpl.formats.folia.Feature class method), parsexml() (pynlpl.formats.folia.Figure class method), parsexml() (pynlpl.formats.folia.Gap class method), 195 parsexml() (pynlpl.formats.folia.Head class method), 207

parsexml() (pynlpl.formats.folia.Headspan class method),

- 792 parsexml() (pynlpl.formats.folia.LangAnnotation class method), 454 parsexml() (pynlpl.formats.folia.LemmaAnnotation class method), 465 parsexml() (pynlpl.formats.folia.Linebreak class method), parsexml() (pynlpl.formats.folia.List class method), 233 parsexml() (pynlpl.formats.folia.ListItem class method), parsexml() (pynlpl.formats.folia.Metric class method), 1046 parsexml() (pynlpl.formats.folia.New class method), 979 parsexml() (pynlpl.formats.folia.Note class method), 259 parsexml() (pynlpl.formats.folia.Observation method), 568 parsexml() (pynlpl.formats.folia.ObservationLayer class method), 698 parsexml() (pynlpl.formats.folia.Original class method), parsexml() (pynlpl.formats.folia.Paragraph class method), parsexml() (pynlpl.formats.folia.Part class method), 285 (pvnlpl.formats.folia.PhonContent parsexml() method), 509 parsexml() (pynlpl.formats.folia.PosAnnotation class method), 443 parsexml() (pynlpl.formats.folia.Predicate class method), parsexml() (pynlpl.formats.folia.Quote class method), parsexml() (pynlpl.formats.folia.Reference class method), parsexml() (pynlpl.formats.folia.Row class method), 324 (pynlpl.formats.folia.SemanticRole parsexml() method), 627 parsexml() (pynlpl.formats.folia.SemanticRolesLayer class method), 745 parsexml() (pynlpl.formats.folia.SenseAnnotation class method), 476 parsexml() (pynlpl.formats.folia.Sentence class method), parsexml() (pynlpl.formats.folia.Sentiment class method), parsexml() (pynlpl.formats.folia.SentimentLayer class method), 710 parsexml() (pynlpl.formats.folia.Statement class method), parsexml() (pynlpl.formats.folia.StatementLayer class method), 721 parsexml() (pynlpl.formats.folia.SubjectivityAnnotation class method), 487 parsexml() (pynlpl.formats.folia.Suggestion class method), 1001
- (pynlpl.formats.folia.SynsetFeature parsexml() class method), 877 (pynlpl.formats.folia.SyntacticUnit parsexml() class method), 615 parsexml() (pynlpl.formats.folia.SyntaxLayer class method), 733 parsexml() (pynlpl.formats.folia.Table class method), 352 (pynlpl.formats.folia.TableHead parsexml() method), 378 parsexml() (pynlpl.formats.folia.Term class method), 365 parsexml() (pynlpl.formats.folia.Text class method), 391 (pynlpl.formats.folia.TextContent parsexml() method), 499 parsexml() (pynlpl.formats.folia.TextMarkupCorrection class method), 844 parsexml() (pynlpl.formats.folia.TextMarkupError class method), 855 parsexml() (pynlpl.formats.folia.TextMarkupGap class method), 812 parsexml() (pynlpl.formats.folia.TextMarkupString class method), 823 parsexml() (pynlpl.formats.folia.TextMarkupStyle class method), 833 (pynlpl.formats.folia.TimeSegment parsexml() class method), 638 parsexml() (pynlpl.formats.folia.TimingLayer class method), 757 (pynlpl.formats.folia.Whitespace parsexml() class method), 404 parsexml() (pynlpl.formats.folia.Word class method), 419 parsexmldeclarations() (pynlpl.formats.folia.Document method), 20 Part (class in pynlpl.formats.folia), 277 path() (pynlpl.datatypes.Trie method), 7 path() (pynlpl.search.AbstractSearchState method), 1060 pathcost() (pynlpl.search.AbstractSearchState method), 1060 Pattern (class in pynlpl.datatypes), 5 PatternMap (class in pynlpl.datatypes), 5 PatternSet (class in pynlpl.datatypes), 5 (pynlpl.formats.folia.Document pendingvalidation() method), 20 perplexity() (pynlpl.statistics.Distribution method), 1062 (pynlpl.formats.folia.AbstractAnnotationLayer phon() method), 79 (pynlpl.formats.folia.AbstractElement method), phon() (pynlpl.formats.folia.AbstractSpanAnnotation phon() method), 56

(pynlpl.formats.folia.AbstractStructureElement

(pynlpl.formats.folia.AbstractTokenAnnotation

(pynlpl.formats.folia.AbstractTextMarkup

1136 Index

phon()

phon()

phon()

method), 42

method), 90

method), 67	phon() (pynlpl.formats.folia.Note method), 259
phon() (pynlpl.formats.folia.ActorFeature method), 889	phon() (pynlpl.formats.folia.Observation method), 569
phon() (pynlpl.formats.folia.Alignment method), 1013	phon() (pynlpl.formats.folia.ObservationLayer method),
phon() (pynlpl.formats.folia.AlignReference method),	698
1024	phon() (pynlpl.formats.folia.Original method), 990
phon() (pynlpl.formats.folia.Alternative method), 924	phon() (pynlpl.formats.folia.Paragraph method), 272
phon() (pynlpl.formats.folia.AlternativeLayers method),	phon() (pynlpl.formats.folia.Part method), 285
935	phon() (pynlpl.formats.folia.PhonContent method), 509
	phon() (pynlpl.formats.folia.PosAnnotation method), 443
phon() (pynlpl.formats.folia.BegindatetimeFeature method), 900	phon() (pynlpl.formats.folia.Predicate method), 580
phon() (pynlpl.formats.folia.Cell method), 106	
	phon() (pynlpl.formats.folia.Quote method), 298
phon() (pynlpl.formats.folia.Chunk method), 522	phon() (pynlpl.formats.folia.Reference method), 311
phon() (pynlpl.formats.folia.ChunkingLayer method),	phon() (pynlpl.formats.folia.Row method), 324
651	phon() (pynlpl.formats.folia.SemanticRole method), 627
phon() (pynlpl.formats.folia.CoreferenceChain method),	phon() (pynlpl.formats.folia.SemanticRolesLayer
534	method), 745
phon() (pynlpl.formats.folia.CoreferenceLayer method),	phon() (pynlpl.formats.folia.SenseAnnotation method),
663	476
phon() (pynlpl.formats.folia.CoreferenceLink method),	phon() (pynlpl.formats.folia.Sentence method), 339
768	phon() (pynlpl.formats.folia.Sentiment method), 592
phon() (pynlpl.formats.folia.Correction method), 948	phon() (pynlpl.formats.folia.SentimentLayer method),
phon() (pynlpl.formats.folia.Current method), 957	710
phon() (pynlpl.formats.folia.Definition method), 119	phon() (pynlpl.formats.folia.Statement method), 603
phon() (pynlpl.formats.folia.DependenciesLayer	phon() (pynlpl.formats.folia.StatementLayer method),
method), 674	721
phon() (pynlpl.formats.folia.Dependency method), 545	phon() (pynlpl.formats.folia.SubjectivityAnnotation
phon() (pynlpl.formats.folia.DependencyDependent	method), 487
method), 780	phon() (pynlpl.formats.folia.Suggestion method), 1001
phon() (pynlpl.formats.folia.Description method), 1035	phon() (pynlpl.formats.folia.SynsetFeature method), 877
phon() (pynlpl.formats.folia.Division method), 131	phon() (pynlpl.formats.folia.SyntacticUnit method), 615
phon() (pynlpl.formats.folia.DomainAnnotation method),	phon() (pynlpl.formats.folia.SyntaxLayer method), 733
432	phon() (pynlpl.formats.folia.Table method), 352
phon() (pynlpl.formats.folia.EnddatetimeFeature	phon() (pynlpl.formats.folia.TableHead method), 378
method), 911	phon() (pynlpl.formats.folia.Term method), 365
phon() (pynlpl.formats.folia.EntitiesLayer method), 686	phon() (pynlpl.formats.folia.Text method), 391
phon() (pynlpl.formats.folia.Entity method), 557	phon() (pynlpl.formats.folia.Text Content method), 499
phon() (pynlpl.formats.folia.Entry method), 144 phon() (pynlpl.formats.folia.ErrorDetection method), 968	phon() (pynlpl.formats.folia.TextMarkupCorrection
	method), 844
phon() (pynlpl.formats.folia.Event method), 157	phon() (pynlpl.formats.folia.TextMarkupError method),
phon() (pynlpl.formats.folia.Example method), 170	855
phon() (pynlpl.formats.folia.Feature method), 866	phon() (pynlpl.formats.folia.TextMarkupGap method),
phon() (pynlpl.formats.folia.Figure method), 184	812
phon() (pynlpl.formats.folia.Gap method), 195	phon() (pynlpl.formats.folia.TextMarkupString method),
phon() (pynlpl.formats.folia.Head method), 208	823
phon() (pynlpl.formats.folia.Headspan method), 792	phon() (pynlpl.formats.folia.TextMarkupStyle method),
phon()  (pynlpl.formats.folia. Lang Annotation  method),	833
454	phon() (pynlpl.formats.folia.TimeSegment method), 638
phon() (pynlpl.formats.folia.LemmaAnnotation method),	phon() (pynlpl.formats.folia.TimingLayer method), 757
465	phon() (pynlpl.formats.folia.Whitespace method), 404
phon() (pynlpl.formats.folia.Linebreak method), 220	phon() (pynlpl.formats.folia.Word method), 419
phon() (pynlpl.formats.folia.List method), 233	$PHONCONTAINER\ (pynlpl. formats. folia. Abstract Annotation Layer$
phon() (pynlpl.formats.folia.ListItem method), 246	attribute), 74
phon() (pynlpl.formats.folia.Metric method), 1046	PHONCONTAINER (pynlpl.formats.folia.AbstractElement
phon() (pynlpl.formats.folia.New method), 979	attribute), 26

attribute), 682

PHONCONTAINER (pynlpl.formats.folia.AbstractSpanAnrattONCONTAINER (pynlpl.formats.folia.Entity attribute), 52 tribute), 553 PHONCONTAINER (pynlpl.formats.folia.AbstractStructure PHIONGONTAINER (pynlpl.formats.folia.Entry attribute), 37 tribute), 139 PHONCONTAINER (pynlpl.formats.folia.AbstractTextMarPHONCONTAINER (pynlpl.formats.folia.ErrorDetection attribute), 86 attribute), 964 PHONCONTAINER (pynlpl.formats.folia.AbstractTokenAnhHatNGONTAINER (pynlpl.formats.folia.Event attribute), 63 tribute), 152 PHONCONTAINER (pynlpl.formats.folia.ActorFeature PHONCONTAINER (pynlpl.formats.folia.Example attribute), 165 attribute), 885 PHONCONTAINER (pynlpl.formats.folia.Alignment at-PHONCONTAINER (pynlpl.formats.folia.Feature tribute), 1009 attribute), 862 (pynlpl.formats.folia.Figure PHONCONTAINER (pynlpl.formats.folia.AlignReference PHONCONTAINER attribute), 1020 tribute), 178 PHONCONTAINER (pynlpl.formats.folia.Alternative at- PHONCONTAINER (pynlpl.formats.folia.Gap attribute), tribute), 919 PHONCONTAINER (pynlpl.formats.folia.AlternativeLayerPHONCONTAINER (pynlpl.formats.folia.Head attribute), 931 tribute), 202 PHONCONTAINER (pynlpl.formats.folia.BegindatetimeFental Menor Container (pynlpl.formats.folia.Headspan atattribute), 896 tribute), 788 PHONCONTAINER (pynlpl.formats.folia.Cell attribute), PHONCONTAINER (pynlpl.formats.folia.LangAnnotation attribute), 450 PHONCONTAINER (pynlpl.formats.folia.Chunk at- PHONCONTAINER (pynlpl.formats.folia.LemmaAnnotation tribute), 518 attribute), 461 PHONCONTAINER (pynlpl.formats.folia.ChunkingLayer PHONCONTAINER (pynlpl.formats.folia.Linebreak atattribute), 646 tribute), 215 PHONCONTAINER (pynlpl.formats.folia.CoreferenceChainPHONCONTAINER (pynlpl.formats.folia.List attribute), attribute), 529 228 PHONCONTAINER (pynlpl.formats.folia.CoreferenceLayePHONCONTAINER (pynlpl.formats.folia.ListItem atattribute), 658 tribute), 241 PHONCONTAINER (pynlpl.formats.folia.CoreferenceLinkPHONCONTAINER (pynlpl.formats.folia.Metric atattribute), 764 tribute), 1042 PHONCONTAINER (pynlpl.formats.folia.Correction at- PHONCONTAINER (pynlpl.formats.folia.New attribute), 944 tribute), 975 (pynlpl.formats.folia.Current (pynlpl.formats.folia.Note PHONCONTAINER PHONCONTAINER attribute), 953 tribute), 254 PHONCONTAINER (pynlpl.formats.folia.Definition at-PHONCONTAINER (pynlpl.formats.folia.Observation tribute), 113 attribute), 564 PHONCONTAINER (pynlpl.formats.folia.DependenciesLavellONCONTAINER (pynlpl.formats.folia.ObservationLayer attribute), 670 attribute), 694 PHONCONTAINER (pynlpl.formats.folia.Dependency PHONCONTAINER (pynlpl.formats.folia.Original atattribute), 541 tribute), 986 PHONCONTAINER (pynlpl.formats.folia.DependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependency attribute), 776 tribute), 267 PHONCONTAINER (pynlpl.formats.folia.Description PHONCONTAINER (pynlpl.formats.folia.Part attribute), attribute), 1031 PHONCONTAINER (pynlpl.formats.folia.Division at PHONCONTAINER (pynlpl.formats.folia.PhonContent tribute), 126 attribute), 506 PHONCONTAINER (pynlpl.formats.folia.DomainAnnotationHONCONTAINER (pynlpl.formats.folia.PosAnnotation attribute), 428 attribute), 439 PHONCONTAINER (pynlpl.formats.folia.EnddatetimeFeattheONCONTAINER (pynlpl.formats.folia.Predicate atattribute), 907 tribute), 576 PHONCONTAINER (pynlpl.formats.folia.EntitiesLayer PHONCONTAINER (pynlpl.formats.folia.Quote

1138 Index

tribute), 293

PHONCONTAINER (pynlpl.formats.folia.Reference at- PHONCONTAINER (pynlpl.formats.folia.Whitespace tribute), 306 attribute), 399 PHONCONTAINER PHONCONTAINER (pynlpl.formats.folia.Word (pynlpl.formats.folia.Row tribute), 319 tribute), 413 PhonContent (class in pynlpl.formats.folia), 503 PHONCONTAINER (pynlpl.formats.folia.SemanticRole attribute), 623 phoncontent() (pynlpl.formats.folia.AbstractAnnotationLayer PHONCONTAINER (pynlpl.formats.folia.SemanticRolesLayer method), 80 attribute), 741 phoncontent() (pynlpl.formats.folia.AbstractElement method), 30 PHONCONTAINER (pynlpl.formats.folia.SenseAnnotation attribute), 472 phoncontent() (pynlpl.formats.folia.AbstractSpanAnnotation PHONCONTAINER (pynlpl.formats.folia.Sentence atmethod), 57 tribute), 333 phoncontent() (pynlpl.formats.folia.AbstractStructureElement PHONCONTAINER (pynlpl.formats.folia.Sentiment atmethod), 43 tribute), 588 phoncontent() (pynlpl.formats.folia.AbstractTextMarkup PHONCONTAINER (pynlpl.formats.folia.SentimentLayer method), 91 attribute), 705 phoncontent() (pynlpl.formats.folia.AbstractTokenAnnotation PHONCONTAINER (pynlpl.formats.folia.Statement atmethod), 68 (pynlpl.formats.folia.ActorFeature tribute), 599 phoncontent() PHONCONTAINER (pynlpl.formats.folia.StatementLayer method), 889 attribute), 717 phoncontent() (pynlpl.formats.folia.Alignment method), PHONCONTAINER (pynlpl.formats.folia.SubjectivityAnnotation 1013 attribute), 483 phoncontent() (pynlpl.formats.folia.AlignReference PHONCONTAINER (pynlpl.formats.folia.Suggestion atmethod), 1025 tribute), 997 phoncontent() (pynlpl.formats.folia.Alternative method), 925 PHONCONTAINER (pynlpl.formats.folia.SynsetFeature attribute), 873 phoncontent() (pynlpl.formats.folia.AlternativeLayers PHONCONTAINER (pynlpl.formats.folia.SyntacticUnit method), 936 attribute), 611 phoncontent() (pynlpl.formats.folia.BegindatetimeFeature PHONCONTAINER (pynlpl.formats.folia.SyntaxLayer method), 900 phoncontent() (pynlpl.formats.folia.Cell method), 106 attribute), 729 phoncontent() (pynlpl.formats.folia.Chunk method), 523 PHONCONTAINER (pynlpl.formats.folia.Table tribute), 347 phoncontent() (pynlpl.formats.folia.ChunkingLayer PHONCONTAINER (pynlpl.formats.folia.TableHead atmethod), 652 tribute), 373 phoncontent() (pynlpl.formats.folia.CoreferenceChain (pynlpl.formats.folia.Term PHONCONTAINER method), 534 tribute), 360 phoncontent() (pynlpl.formats.folia.CoreferenceLayer PHONCONTAINER (pynlpl.formats.folia.Text attribute), method), 663 phoncontent() (pynlpl.formats.folia.CoreferenceLink PHONCONTAINER (pynlpl.formats.folia.TextContent method), 769 phoncontent() (pynlpl.formats.folia.Correction method), attribute), 495 PHONCONTAINER (pynlpl.formats.folia.TextMarkupCorrection attribute), 840 phoncontent() (pynlpl.formats.folia.Current method), 958 PHONCONTAINER (pynlpl.formats.folia.TextMarkupErrorphoncontent() (pynlpl.formats.folia.Definition method), attribute), 851 PHONCONTAINER (pynlpl.formats.folia.TextMarkupGap phoncontent() (pynlpl.formats.folia.DependenciesLayer method), 675 attribute), 808 PHONCONTAINER (pynlpl.formats.folia.TextMarkupStringhoncontent() (pynlpl.formats.folia.Dependency attribute), 819 method), 546 PHONCONTAINER (pynlpl.formats.folia.TextMarkupStylephoncontent() (pynlpl.formats.folia.DependencyDependent attribute), 830 method), 781 PHONCONTAINER (pynlpl.formats.folia.TimeSegment phoncontent() (pynlpl.formats.folia.Description method),

Index 1139

PHONCONTAINER (pynlpl.formats.folia.TimingLayer phoncontent() (pynlpl.formats.folia.Division method),

1036

132

attribute), 634

attribute), 752

phoncontent() (pynlpl.formats.folia.DomainAnnotation method), 433 phoncontent() (pynlpl.formats.folia.EnddatetimeFeature method), 911 (pynlpl.formats.folia.EntitiesLayer phoncontent() method), 687 phoncontent() (pynlpl.formats.folia.Entity method), 558 phoncontent() (pynlpl.formats.folia.Entry method), 145 phoncontent() (pynlpl.formats.folia.ErrorDetection method), 969 phoncontent() (pynlpl.formats.folia.Event method), 158 phoncontent() (pynlpl.formats.folia.Example method), phoncontent() (pynlpl.formats.folia.Feature method), 867 phoncontent() (pynlpl.formats.folia.Figure method), 184 phoncontent() (pynlpl.formats.folia.Gap method), 195 phoncontent() (pynlpl.formats.folia.Head method), 208 phoncontent() (pynlpl.formats.folia.Headspan method), 793 (pynlpl.formats.folia.LangAnnotation phoncontent() method), 455 phoncontent() (pynlpl.formats.folia.LemmaAnnotation method), 466 phoncontent() (pynlpl.formats.folia.Linebreak method), 221 phoncontent() (pynlpl.formats.folia.List method), 234 phoncontent() (pynlpl.formats.folia.ListItem method), phoncontent() (pynlpl.formats.folia.Metric method), 1047 phoncontent() (pynlpl.formats.folia.New method), 980 phoncontent() (pynlpl.formats.folia.Note method), 260 phoncontent() (pynlpl.formats.folia.Observation method), 569 phoncontent() (pynlpl.formats.folia.ObservationLayer method), 699 phoncontent() (pynlpl.formats.folia.Original method), 991 phoncontent() (pynlpl.formats.folia.Paragraph method), 273 phoncontent() (pynlpl.formats.folia.Part method), 286 phoncontent() (pynlpl.formats.folia.PhonContent method), 509 (pynlpl.formats.folia.PosAnnotation phoncontent() method), 444 phoncontent() (pynlpl.formats.folia.Predicate method), 581 phoncontent() (pynlpl.formats.folia.Quote method), 299 phoncontent() (pynlpl.formats.folia.Reference method), phoncontent() (pynlpl.formats.folia.Row method), 325 (pynlpl.formats.folia.SemanticRole phoncontent() method), 628 phoncontent() (pynlpl.formats.folia.SemanticRolesLayer

method), 746

phoncontent() (pynlpl.formats.folia.SenseAnnotation method), 477 phoncontent() (pynlpl.formats.folia.Sentence method), 340 phoncontent() (pynlpl.formats.folia.Sentiment method), phoncontent() (pynlpl.formats.folia.SentimentLayer method), 710 phoncontent() (pynlpl.formats.folia.Statement method), 604 phoncontent() (pynlpl.formats.folia.StatementLayer method), 722 phoncontent() (pynlpl.formats.folia.SubjectivityAnnotation method), 488 phoncontent() (pynlpl.formats.folia.Suggestion method), 1002 phoncontent() (pynlpl.formats.folia.SynsetFeature method), 878 phoncontent() (pynlpl.formats.folia.SyntacticUnit method), 616 phoncontent() (pynlpl.formats.folia.SyntaxLayer method), 734 phoncontent() (pynlpl.formats.folia.Table method), 353 phoncontent() (pynlpl.formats.folia.TableHead method), 379 phoncontent() (pynlpl.formats.folia.Term method), 366 phoncontent() (pynlpl.formats.folia.Text method), 392 phoncontent() (pynlpl.formats.folia.TextContent method), 499 phoncontent() (pynlpl.formats.folia.TextMarkupCorrection method), 845 phoncontent() (pynlpl.formats.folia.TextMarkupError method), 855 phoncontent() (pynlpl.formats.folia.TextMarkupGap method), 813 phoncontent() (pynlpl.formats.folia.TextMarkupString method), 823 phoncontent() (pynlpl.formats.folia.TextMarkupStyle method), 834 phoncontent() (pynlpl.formats.folia.TimeSegment method), 639 phoncontent() (pynlpl.formats.folia.TimingLayer method), 757 phoncontent() (pynlpl.formats.folia.Whitespace method), phoncontent() (pynlpl.formats.folia.Word method), 420 phoneme() (pynlpl.formats.folia.Word method), 420 phonemes() (pynlpl.formats.folia.Word method), 420 PhraseTable (class in pynlpl.formats.moses), 1054 PhraseTableClient (class in pynlpl.formats.moses), 1054 poll() (pynlpl.evaluation.ExperimentPool method), 10 pop() (pynlpl.datatypes.FIFOQueue method), 5

pop() (pynlpl.datatypes.PriorityQueue method), 6

pos() (pynlpl.formats.folia.Word method), 420

PosAnnotation (class in pynlpl.formats.folia), 437 postappend() (pynlpl.formats.folia.EntitiesLayer poslog() (pynlpl.statistics.Distribution method), 1062 method), 687 postappend() (pynlpl.formats.folia.AbstractAnnotationLayerpostappend() (pynlpl.formats.folia.Entity method), 558 method), 80 postappend() (pynlpl.formats.folia.Entry method), 146 (pynlpl.formats.folia.ErrorDetection postappend() (pynlpl.formats.folia.AbstractElement postappend() method), 31 method), 969 postappend() (pynlpl.formats.folia.AbstractSpanAnnotation postappend() (pynlpl.formats.folia.Event method), 159 postappend() (pynlpl.formats.folia.Example method), 172 method), 57 postappend() (pynlpl.formats.folia.AbstractStructureElemenpostappend() (pynlpl.formats.folia.Feature method), 867 method), 44 postappend() (pynlpl.formats.folia.Figure method), 185 postappend() (pynlpl.formats.folia.AbstractTextMarkup postappend() (pynlpl.formats.folia.Gap method), 196 method), 91 postappend() (pynlpl.formats.folia.Head method), 209 postappend() (pynlpl.formats.folia.AbstractTokenAnnotatiorpostappend() (pynlpl.formats.folia.Headspan method), 793 method), 68 postappend() (pynlpl.formats.folia.ActorFeature method), postappend() (pynlpl.formats.folia.LangAnnotation method), 455 postappend() (pynlpl.formats.folia.Alignment method), (pynlpl.formats.folia.LemmaAnnotation postappend() 1014 method), 466 postappend() (pynlpl.formats.folia.Linebreak method), postappend() (pynlpl.formats.folia.AlignReference method), 1025 postappend() (pynlpl.formats.folia.Alternative method), postappend() (pynlpl.formats.folia.List method), 235 postappend() (pynlpl.formats.folia.ListItem method), 248 (pynlpl.formats.folia.AlternativeLayers postappend() (pynlpl.formats.folia.Metric method), 1047 postappend() method), 936 postappend() (pynlpl.formats.folia.New method), 980 postappend() (pynlpl.formats.folia.BegindatetimeFeature postappend() (pynlpl.formats.folia.Note method), 261 method), 901 postappend() (pynlpl.formats.folia.Observation method), postappend() (pynlpl.formats.folia.Cell method), 107 570 postappend() (pynlpl.formats.folia.Chunk method), 523 postappend() (pynlpl.formats.folia.ObservationLayer (pynlpl.formats.folia.ChunkingLayer method), 699 postappend() postappend() (pynlpl.formats.folia.Original method), 991 method), 652 (pynlpl.formats.folia.CoreferenceChain postappend() (pynlpl.formats.folia.Paragraph method), postappend() method), 535 274 (pynlpl.formats.folia.CoreferenceLayer postappend() (pynlpl.formats.folia.Part method), 287 postappend() method), 664 postappend() (pynlpl.formats.folia.PhonContent method), (pynlpl.formats.folia.CoreferenceLink postappend() 510 method), 769 postappend() (pynlpl.formats.folia.PosAnnotation postappend() (pynlpl.formats.folia.Correction method), method), 444 postappend() (pynlpl.formats.folia.Predicate method), postappend() (pynlpl.formats.folia.Current method), 958 581 postappend() (pynlpl.formats.folia.Definition method), postappend() (pynlpl.formats.folia.Quote method), 300 postappend() (pynlpl.formats.folia.Reference method), postappend() (pynlpl.formats.folia.DependenciesLayer method), 676 postappend() (pynlpl.formats.folia.Row method), 326 postappend() (pynlpl.formats.folia.Dependency method), postappend() (pynlpl.formats.folia.SemanticRole method), 628 postappend() (pynlpl.formats.folia.DependencyDependent postappend() (pynlpl.formats.folia.SemanticRolesLayer method), 781 method), 746 postappend() (pynlpl.formats.folia.Description method), (pynlpl.formats.folia.SenseAnnotation postappend() method), 477 postappend() (pynlpl.formats.folia.Division method), 133 (pynlpl.formats.folia.Sentence method), postappend() (pynlpl.formats.folia.DomainAnnotation 340 postappend() postappend() (pynlpl.formats.folia.Sentiment method), postappend() (pynlpl.formats.folia.EnddatetimeFeature 593 method), 912 (pynlpl.formats.folia.SentimentLayer postappend()

method), 711 postappend() (pynlpl.formats.folia.Statement method), 605 (pynlpl.formats.folia.StatementLayer postappend() method), 723 postappend() (pynlpl.formats.folia.SubjectivityAnnotation method), 488 postappend() (pynlpl.formats.folia.Suggestion method), 1002 (pynlpl.formats.folia.SynsetFeature postappend() method), 879 (pynlpl.formats.folia.SyntacticUnit postappend() method), 616 postappend() (pynlpl.formats.folia.SyntaxLayer method), 734 postappend() (pynlpl.formats.folia.Table method), 353 postappend() (pynlpl.formats.folia.TableHead method), postappend() (pynlpl.formats.folia.Term method), 366 postappend() (pynlpl.formats.folia.Text method), 392 postappend() (pynlpl.formats.folia.TextContent method), postappend() (pynlpl.formats.folia.TextMarkupCorrection method), 845 (pynlpl.formats.folia.TextMarkupError postappend() method), 856 postappend() (pynlpl.formats.folia.TextMarkupGap method), 813 (pynlpl.formats.folia.TextMarkupString postappend() method), 824 postappend() (pynlpl.formats.folia.TextMarkupStyle method), 834 (pynlpl.formats.folia.TimeSegment postappend() method), 640 postappend() (pynlpl.formats.folia.TimingLayer method), postappend() (pynlpl.formats.folia.Whitespace method), postappend() (pynlpl.formats.folia.Word method), 420 precision() (pynlpl.evaluation.ClassEvaluation method), Predicate (class in pynlpl.formats.folia), 573 previous() (pynlpl.formats.folia.AbstractAnnotationLayer method), 80 (pynlpl.formats.folia.AbstractElement previous() method), 31 previous() (pynlpl.formats.folia.AbstractSpanAnnotation method), 57 previous() (pynlpl.formats.folia.AbstractStructureElement method), 44 (pynlpl.formats.folia.AbstractTextMarkup previous() method), 91 previous() (pynlpl.formats.folia.AbstractTokenAnnotation previous()

method), 68

previous() (pynlpl.formats.folia.ActorFeature method), (pynlpl.formats.folia.Alignment previous() previous() (pynlpl.formats.folia.AlignReference method), previous() (pynlpl.formats.folia.Alternative method), 925 (pynlpl.formats.folia.AlternativeLayers previous() method), 936 (pynlpl.formats.folia.BegindatetimeFeature previous() method), 901 previous() (pynlpl.formats.folia.Cell method), 107 previous() (pynlpl.formats.folia.Chunk method), 523 previous() (pynlpl.formats.folia.ChunkingLayer method), previous() (pynlpl.formats.folia.CoreferenceChain method), 535 previous() (pynlpl.formats.folia.CoreferenceLayer method), 664 (pynlpl.formats.folia.CoreferenceLink previous() method), 770 previous() (pynlpl.formats.folia.Correction method), 948 previous() (pynlpl.formats.folia.Current method), 958 previous() (pynlpl.formats.folia.Definition method), 120 previous() (pynlpl.formats.folia.DependenciesLayer method), 676 previous() (pynlpl.formats.folia.Dependency method), (pynlpl.formats.folia.DependencyDependent previous() method), 781 (pynlpl.formats.folia.Description method), previous() 1036 previous() (pynlpl.formats.folia.Division method), 133 (pynlpl.formats.folia.DomainAnnotation previous() method), 433 (pynlpl.formats.folia.EnddatetimeFeature previous() method), 912 previous() (pynlpl.formats.folia.EntitiesLayer method), previous() (pynlpl.formats.folia.Entity method), 558 previous() (pynlpl.formats.folia.Entry method), 146 previous() (pynlpl.formats.folia.ErrorDetection method), previous() (pynlpl.formats.folia.Event method), 159 previous() (pynlpl.formats.folia.Example method), 172 previous() (pynlpl.formats.folia.Feature method), 867 previous() (pynlpl.formats.folia.Figure method), 185 previous() (pynlpl.formats.folia.Gap method), 196 previous() (pynlpl.formats.folia.Head method), 209 previous() (pynlpl.formats.folia.Headspan method), 793 previous() (pynlpl.formats.folia.LangAnnotation

1142 Index

method), 455

method), 466

(pynlpl.formats.folia.LemmaAnnotation

previous() (pynlpl.formats.folia.Linebreak method), 222 previous() (pynlpl.formats.folia.TextMarkupGap previous() (pynlpl.formats.folia.List method), 235 method), 813 previous() (pynlpl.formats.folia.ListItem method), 248 (pynlpl.formats.folia.TextMarkupString previous() previous() (pynlpl.formats.folia.Metric method), 1047 method), 824 previous() (pynlpl.formats.folia.New method), 980 previous() (pynlpl.formats.folia.TextMarkupStyle previous() (pynlpl.formats.folia.Note method), 261 method), 834 previous() (pynlpl.formats.folia.Observation method), previous() (pynlpl.formats.folia.TimeSegment method), 570 previous() (pynlpl.formats.folia.TimingLayer method), previous() (pynlpl.formats.folia.ObservationLayer method), 699 previous() (pynlpl.formats.folia.Original method), 991 previous() (pynlpl.formats.folia.Whitespace method), 405 previous() (pynlpl.formats.folia.Paragraph method), 274 previous() (pynlpl.formats.folia.Word method), 420 previous() (pynlpl.formats.folia.Part method), 287 PRIMARYELEMENT (pynlpl.formats.folia.AbstractAnnotationLayer previous() (pynlpl.formats.folia.PhonContent method), attribute), 75 510 PRIMARYELEMENT (pynlpl.formats.folia.AbstractElement previous() (pynlpl.formats.folia.PosAnnotation method), attribute), 26 PRIMARYELEMENT (pynlpl.formats.folia.AbstractSpanAnnotation previous() (pynlpl.formats.folia.Predicate method), 581 attribute), 52 previous() (pynlpl.formats.folia.Quote method), 300 PRIMARYELEMENT (pynlpl.formats.folia.AbstractStructureElement previous() (pynlpl.formats.folia.Reference method), 313 attribute), 37 previous() (pynlpl.formats.folia.Row method), 326 PRIMARYELEMENT (pynlpl.formats.folia.AbstractTextMarkup previous() (pynlpl.formats.folia.SemanticRole method), attribute), 86 628 PRIMARYELEMENT (pynlpl. formats. folia. Abstract Token Annotationprevious() (pynlpl.formats.folia.SemanticRolesLayer attribute), 63 method), 746 PRIMARYELEMENT (pynlpl.formats.folia.ActorFeature previous() (pynlpl.formats.folia.SenseAnnotation attribute), 885 method), 477 PRIMARYELEMENT (pynlpl.formats.folia.Alignment previous() (pynlpl.formats.folia.Sentence method), 340 attribute), 1009 previous() (pynlpl.formats.folia.Sentiment method), 593 PRIMARYELEMENT (pynlpl.formats.folia.AlignReference previous() (pynlpl.formats.folia.SentimentLayer method), attribute), 1020 PRIMARYELEMENT (pynlpl.formats.folia.Alternative previous() (pynlpl.formats.folia.Statement method), 605 attribute), 919 previous() (pynlpl.formats.folia.StatementLayer method), PRIMARYELEMENT (pynlpl.formats.folia.AlternativeLayers attribute), 931 previous() (pynlpl.formats.folia.SubjectivityAnnotation PRIMARYELEMENT (pynlpl.formats.folia.BegindatetimeFeature method), 488 attribute), 896 previous() (pynlpl.formats.folia.Suggestion method), **PRIMARYELEMENT** (pynlpl.formats.folia.Cell tribute), 100 previous() (pynlpl.formats.folia.SynsetFeature method), PRIMARYELEMENT (pynlpl.formats.folia.Chunk attribute), 518 previous() (pynlpl.formats.folia.SyntacticUnit method), PRIMARYELEMENT (pynlpl.formats.folia.ChunkingLayer attribute), 646 previous() (pynlpl.formats.folia.SyntaxLayer method), PRIMARYELEMENT (pynlpl.formats.folia.CoreferenceChain attribute), 529 previous() (pynlpl.formats.folia.Table method), 353 PRIMARYELEMENT (pynlpl.formats.folia.CoreferenceLayer previous() (pynlpl.formats.folia.TableHead method), 379 attribute), 658 previous() (pynlpl.formats.folia.Term method), 366 PRIMARYELEMENT (pynlpl.formats.folia.CoreferenceLink previous() (pynlpl.formats.folia.Text method), 392 attribute), 764 previous() (pynlpl.formats.folia.TextContent method), PRIMARYELEMENT (pynlpl.formats.folia.Correction 500 attribute), 944 (pynlpl.formats.folia.TextMarkupCorrection PRIMARYELEMENT (pynlpl.formats.folia.Current atprevious() method), 845 tribute), 953 previous() (pynlpl.formats.folia.TextMarkupError PRIMARYELEMENT (pynlpl.formats.folia.Definition

Index 1143

attribute), 113

method), 856

```
PRIMARYELEMENT (pynlpl.formats.folia.DependenciesLPRMMARYELEMENT (pynlpl.formats.folia.ObservationLayer
         attribute), 670
                                                              attribute), 694
PRIMARYELEMENT (pynlpl.formats.folia.Dependency PRIMARYELEMENT (pynlpl.formats.folia.Original at-
         attribute), 541
                                                              tribute), 986
PRIMARYELEMENT (pynlpl.formats.folia.DependencyDepartmentyPellement
                                                                            (pynlpl.formats.folia.Paragraph
        attribute), 776
                                                              attribute), 267
PRIMARYELEMENT (pynlpl.formats.folia.Description PRIMARYELEMENT
                                                                             (pynlpl.formats.folia.Part
         attribute), 1031
                                                              tribute), 280
PRIMARYELEMENT (pynlpl.formats.folia.Division at- PRIMARYELEMENT (pynlpl.formats.folia.PhonContent
                                                              attribute), 506
         tribute), 126
PRIMARYELEMENT (pynlpl.formats.folia.DomainAnnotaPRiMARYELEMENT (pynlpl.formats.folia.PosAnnotation
                                                              attribute), 439
         attribute), 428
PRIMARYELEMENT (pynlpl.formats.folia.EnddatetimeFe PRIMARYELEMENT (pynlpl.formats.folia.Predicate at-
         attribute), 907
                                                              tribute), 576
PRIMARYELEMENT (pynlpl.formats.folia.EntitiesLayer PRIMARYELEMENT
                                                                                (pynlpl.formats.folia.Quote
         attribute), 682
                                                              attribute), 293
PRIMARYELEMENT
                           (pynlpl.formats.folia.Entity
                                                     PRIMARYELEMENT
                                                                            (pynlpl.formats.folia.Reference
         attribute), 553
                                                              attribute), 306
                           (pynlpl.formats.folia.Entry
PRIMARYELEMENT
                                                     PRIMARYELEMENT
                                                                            (pynlpl.formats.folia.Row
         attribute), 139
                                                              tribute), 319
PRIMARYELEMENT (pynlpl.formats.folia.ErrorDetection PRIMARYELEMENT (pynlpl.formats.folia.SemanticRole
         attribute), 964
                                                              attribute), 623
PRIMARYELEMENT
                           (pynlpl.formats.folia.Event PRIMARYELEMENT (pynlpl.formats.folia.SemanticRolesLayer
         attribute), 152
                                                              attribute), 741
PRIMARYELEMENT (pynlpl.formats.folia.Example at-
                                                     PRIMARYELEMENT (pynlpl.formats.folia.SenseAnnotation
         tribute), 165
                                                              attribute), 472
PRIMARYELEMENT
                     (pynlpl.formats.folia.Feature at-
                                                     PRIMARYELEMENT (pynlpl.formats.folia.Sentence at-
         tribute), 862
                                                              tribute), 333
PRIMARYELEMENT
                      (pynlpl.formats.folia.Figure at-
                                                     PRIMARYELEMENT
                                                                            (pynlpl.formats.folia.Sentiment
         tribute), 178
                                                              attribute), 588
PRIMARYELEMENT
                       (pynlpl.formats.folia.Gap
                                                at-
                                                     PRIMARYELEMENT (pynlpl.formats.folia.SentimentLayer
         tribute), 191
                                                              attribute), 705
                      (pynlpl.formats.folia.Head
                                                     PRIMARYELEMENT
                                                                            (pynlpl.formats.folia.Statement
PRIMARYELEMENT
         tribute), 202
                                                              attribute), 599
PRIMARYELEMENT
                       (pynlpl.formats.folia.Headspan
                                                     PRIMARYELEMENT (pynlpl.formats.folia.StatementLayer
        attribute), 788
                                                              attribute), 717
PRIMARYELEMENT (pynlpl.formats.folia.LangAnnotation PRIMARYELEMENT (pynlpl.formats.folia.SubjectivityAnnotation
         attribute), 450
                                                              attribute), 483
PRIMARYELEMENT (pynlpl.formats.folia.LemmaAnnotathMMARYELEMENT (pynlpl.formats.folia.Suggestion
         attribute), 461
                                                              attribute), 997
PRIMARYELEMENT
                       (pynlpl.formats.folia.Linebreak
                                                     PRIMARYELEMENT (pynlpl.formats.folia.SynsetFeature
         attribute), 215
                                                              attribute), 873
PRIMARYELEMENT
                       (pynlpl.formats.folia.List
                                                     PRIMARYELEMENT (pynlpl.formats.folia.SyntacticUnit
                                                at-
                                                              attribute), 611
        tribute), 228
PRIMARYELEMENT (pynlpl.formats.folia.ListItem at-
                                                     PRIMARYELEMENT (pynlpl.formats.folia.SyntaxLayer
         tribute), 241
                                                              attribute), 729
PRIMARYELEMENT (pynlpl.formats.folia.Metric at-
                                                     PRIMARYELEMENT
                                                                                (pynlpl.formats.folia.Table
         tribute), 1042
                                                              attribute), 347
                       (pynlpl.formats.folia.New
                                                                            (pynlpl.formats.folia.TableHead
PRIMARYELEMENT
                                                at-
                                                     PRIMARYELEMENT
         tribute), 975
                                                              attribute), 373
PRIMARYELEMENT
                       (pynlpl.formats.folia.Note
                                                     PRIMARYELEMENT
                                                                            (pynlpl.formats.folia.Term
                                                at-
        tribute), 254
                                                              tribute), 360
PRIMARYELEMENT (pynlpl.formats.folia.Observation
                                                    PRIMARYELEMENT
                                                                            (pynlpl.formats.folia.Text
         attribute), 564
                                                              tribute), 386
```

- PRIMARYELEMENT (pynlpl.formats.folia,TextContent PRINTABLE (pynlpl.formats.folia,CoreferenceLink atattribute), 495 tribute), 764
- PRIMARYELEMENT (pynlpl.formats.folia.TextMarkupContraction attribute), attribute), 840
- PRIMARYELEMENT (pynlpl.formats.folia.TextMarkupErrBRINTABLE (pynlpl.formats.folia.Current attribute), 953 attribute), 851 PRINTABLE (pynlpl.formats.folia.Definition attribute),
- PRIMARYELEMENT (pynlpl.formats.folia.TextMarkupGap 113 attribute), 808
- PRIMARYELEMENT (pynlpl.formats.folia.TextMarkupString attribute), 670
- attribute), 819 tribute), 541
- PRIMARYELEMENT (pynlpl.formats.folia.TextMarkupStyle attribute), 830
- PRIMARYELEMENT (pynlpl.formats.folia.TimeSegment attribute), 634
- PRIMARYELEMENT (pynlpl.formats.folia.TimingLayer attribute), 752
- PRIMARYELEMENT (pynlpl.formats.folia.Whitespace attribute), 399
- PRIMARYELEMENT (pynlpl.formats.folia.Word attribute), 413
- print dptable() (pynlpl.statistics.HiddenMarkovModel method), 1063
- PRINTABLE (pynlpl.formats.folia.AbstractAnnotationLayer attribute), 75
- PRINTABLE (pynlpl.formats.folia.AbstractElement attribute), 26
- PRINTABLE (pynlpl.formats.folia.AbstractSpanAnnotation
- attribute), 52
- PRINTABLE (pynlpl.formats.folia.AbstractTextMarkup attribute), 86

attribute), 37

- PRINTABLE (pynlpl.formats.folia.AbstractTokenAnnotationRINTABLE (pynlpl.formats.folia.Gap attribute), 191 attribute), 63
- (pynlpl.formats.folia.ActorFeature **PRINTABLE** tribute), 885
- PRINTABLE (pynlpl.formats.folia.Alignment attribute),
- PRINTABLE (pynlpl.formats.folia.AlignReference attribute), 1020
- PRINTABLE (pynlpl.formats.folia.Alternative attribute),
- PRINTABLE (pynlpl.formats.folia.AlternativeLayers attribute), 931
- PRINTABLE (pynlpl.formats.folia.BegindatetimeFeature attribute), 896
- PRINTABLE (pynlpl.formats.folia.Cell attribute), 100
- PRINTABLE (pynlpl.formats.folia.Chunk attribute), 518
- PRINTABLE (pynlpl.formats.folia.ChunkingLayer attribute), 646
- PRINTABLE (pynlpl.formats.folia.CoreferenceChain attribute), 529
- PRINTABLE (pynlpl.formats.folia.CoreferenceLayer attribute), 658

- - PRINTABLE (pynlpl.formats.folia.DependenciesLayer
- **PRINTABLE** (pynlpl.formats.folia.Dependency
- PRINTABLE (pynlpl.formats.folia.DependencyDependent
- attribute), 776 PRINTABLE (pynlpl.formats.folia.Description attribute), 1031
- PRINTABLE (pynlpl.formats.folia.Division attribute), 126
- **PRINTABLE** (pynlpl.formats.folia.DomainAnnotation attribute), 428
- PRINTABLE (pynlpl.formats.folia.EnddatetimeFeature attribute), 907
- **PRINTABLE** (pynlpl.formats.folia.EntitiesLayer tribute), 682
- PRINTABLE (pynlpl.formats.folia.Entity attribute), 553
- PRINTABLE (pynlpl.formats.folia.Entry attribute), 139
- **PRINTABLE** (pynlpl.formats.folia.ErrorDetection attribute), 964
- PRINTABLE (pynlpl.formats.folia.Event attribute), 152
- PRINTABLE (pynlpl.formats.folia.AbstractStructureElemenRRINTABLE (pynlpl.formats.folia.Example attribute),
  - PRINTABLE (pynlpl.formats.folia.Feature attribute), 862 PRINTABLE (pynlpl.formats.folia.Figure attribute), 178

  - PRINTABLE (pynlpl.formats.folia.Head attribute), 202
  - PRINTABLE (pynlpl.formats.folia.Headspan attribute),
  - 788
  - PRINTABLE (pynlpl.formats.folia.LangAnnotation attribute), 451
  - PRINTABLE (pynlpl.formats.folia.LemmaAnnotation attribute), 462
  - PRINTABLE (pynlpl.formats.folia.Linebreak attribute), 215
  - PRINTABLE (pynlpl.formats.folia.List attribute), 228
  - PRINTABLE (pynlpl.formats.folia.ListItem attribute), 241
  - PRINTABLE (pynlpl.formats.folia.Metric attribute),
  - PRINTABLE (pynlpl.formats.folia.New attribute), 975
  - PRINTABLE (pynlpl.formats.folia.Note attribute), 254
  - **PRINTABLE** (pynlpl.formats.folia.Observation tribute), 564
  - PRINTABLE (pynlpl.formats.folia.ObservationLayer attribute), 694
  - PRINTABLE (pynlpl.formats.folia.Original attribute),

986	tribute), 819
PRINTABLE (pynlpl.formats.folia.Paragraph attribute), 267	PRINTABLE (pynlpl.formats.folia.TextMarkupStyle attribute), 830
PRINTABLE (pynlpl.formats.folia.Part attribute), 280 PRINTABLE (pynlpl.formats.folia.PhonContent at-	PRINTABLE (pynlpl.formats.folia.TimeSegment attribute), 634
tribute), 506	PRINTABLE (pynlpl.formats.folia.TimingLayer attribute), 752
PRINTABLE (pynlpl.formats.folia.PosAnnotation attribute), 440	PRINTABLE (pynlpl.formats.folia.Whitespace attribute),
PRINTABLE (pynlpl.formats.folia.Predicate attribute),	399
576	PRINTABLE (pynlpl.formats.folia.Word attribute), 413
PRINTABLE (pynlpl.formats.folia.Quote attribute), 293	PriorityQueue (class in pynlpl.datatypes), 5
PRINTABLE (pynlpl.formats.folia.Reference attribute), 306	prob() (pynlpl.lm.lm.ARPALanguageModel.NgramsProbs method), 1057
PRINTABLE (pynlpl.formats.folia.Row attribute), 319	ProcessFailed, 10
PRINTABLE (pynlpl.formats.folia.SemanticRole at-	product() (in module pynlpl.statistics), 1064
tribute), 623	prune() (pynlpl.datatypes.PriorityQueue method), 6
PRINTABLE (pynlpl.formats.folia.SemanticRolesLayer	prune() (pynlpl.search.AbstractSearch method), 1059
attribute), 741	prune() (pynlpl.search.BeamedBestFirstSearch method),
PRINTABLE (pynlpl.formats.folia.SenseAnnotation at-	1060
tribute), 473	$prune () \ (pynlpl.search.Early Eager Beam Search \ method),$
PRINTABLE (pynlpl.formats.folia.Sentence attribute),	1060
333	prune() (pynlpl.search.StochasticBeamSearch method),
PRINTABLE (pynlpl.formats.folia.Sentiment attribute), 588	1060
PRINTABLE (pynlpl.formats.folia.SentimentLayer at-	prunebyscore() (pynlpl.datatypes.PriorityQueue method),
tribute), 705	publisher() (pynlpl.formats.folia.Document method), 20
PRINTABLE (pynlpl.formats.folia.Statement attribute),	pynlpl.common (module), 3
599	pynlpl.datatypes (module), 5
PRINTABLE (pynlpl.formats.folia.StatementLayer at-	pynlpl.evaluation (module), 9
tribute), 717	pynlpl.formats.cgn (module), 1053
PRINTABLE(pynlpl.formats.folia. Subjectivity Annotation	pynlpl.formats.folia (module), 11
attribute), 484	pynlpl.formats.giza (module), 1053
PRINTABLE (pynlpl.formats.folia.Suggestion attribute),	pynlpl.formats.moses (module), 1054
997	pynlpl.formats.sonar (module), 1054
PRINTABLE (pynlpl.formats.folia.SynsetFeature at-	pynlpl.formats.taggerdata (module), 1055
tribute), 874 PRINTABLE (pynlpl.formats.folia.SyntacticUnit at-	pynlpl.formats.timbl (module), 1055 pynlpl.lm.client (module), 1058
tribute), 611	pynlpl.lm.lm (module), 1057
PRINTABLE (pynlpl.formats.folia.SyntaxLayer at-	pynlpl.lm.srilm (module), 1057
tribute), 729	pynlpl.search (module), 1059
PRINTABLE (pynlpl.formats.folia.Table attribute), 347	pynlpl.statistics (module), 1062
PRINTABLE (pynlpl.formats.folia.TableHead attribute),	pynlpl.textprocessors (module), 1066
373	
PRINTABLE (pynlpl.formats.folia.Term attribute), 360	Q
PRINTABLE (pynlpl.formats.folia.Text attribute), 386	Query (class in pynlpl.formats.fql), 802
PRINTABLE (pynlpl.formats.folia.TextContent at-	Queue (class in pynlpl.datatypes), 6
tribute), 495	Quote (class in pynlpl.formats.folia), 290
PRINTABLE (pynlpl.formats.folia.TextMarkupCorrection attribute), 840	R
PRINTABLE (pynlpl.formats.folia.TextMarkupError at-	
tribute), 851	randomprune() (pynlpl.datatypes.PriorityQueue method),
PRINTABLE (pynlpl.formats.folia.TextMarkupGap at-	6  Pandar (class in numbral formats falia) 804
tribute), 808	Reader (class in pynlpl.formats.folia), 804 recall() (pynlpl.evaluation.ClassEvaluation method), 10
PRINTABLE (pynlpl.formats.folia.TextMarkupString at-	reducible() (pynlpl.statistics.MarkovChain method), 1063

- Reference (class in pynlpl.formats.folia), 303
- ReflowText (class in pynlpl.textprocessors), 1066
- relaxng() (pynlpl.formats.folia.AbstractAnnotationLayer class method), 80
- relaxng() (pynlpl.formats.folia.AbstractElement class method), 31
- relaxng() (pynlpl.formats.folia.AbstractSpanAnnotation class method), 57
- relaxng() (pynlpl.formats.folia.AbstractStructureElement class method), 44
- relaxng() (pynlpl.formats.folia.AbstractTextMarkup class method), 91
- relaxng() (pynlpl.formats.folia.AbstractTokenAnnotation class method), 68
- relaxng() (pynlpl.formats.folia.ActorFeature class method), 890
- relaxng() (pynlpl.formats.folia.Alignment class method), 1014
- relaxng() (pynlpl.formats.folia.AlignReference class method), 1025
- relaxng() (pynlpl.formats.folia.Alternative class method), 925
- relaxng() (pynlpl.formats.folia.AlternativeLayers class method), 936
- relaxng() (pynlpl.formats.folia.BegindatetimeFeature class method), 901
- relaxng() (pynlpl.formats.folia.Cell class method), 107
- relaxng() (pynlpl.formats.folia.Chunk class method), 523
- relaxng() (pynlpl.formats.folia.ChunkingLayer class method), 652
- relaxng() (pynlpl.formats.folia.CoreferenceChain class method), 535
- relaxng() (pynlpl.formats.folia.CoreferenceLayer class method), 664
- relaxng() (pynlpl.formats.folia.CoreferenceLink class method), 770
- relaxng() (pynlpl.formats.folia.Correction class method),
- relaxng() (pynlpl.formats.folia.Current class method), 959
- relaxng() (pynlpl.formats.folia.Definition class method), 120
- relaxng() (pynlpl.formats.folia.DependenciesLayer class method), 676
- relaxng() (pynlpl.formats.folia.Dependency class method), 547
- relaxng() (pynlpl.formats.folia.DependencyDependent class method), 782
- relaxng() (pynlpl.formats.folia.Description class method),
- relaxng() (pynlpl.formats.folia.Division class method),
- relaxng() (pynlpl.formats.folia.DomainAnnotation class method), 434

- relaxng() (pynlpl.formats.folia.EnddatetimeFeature class method), 912
- relaxng() (pynlpl.formats.folia.EntitiesLayer class method), 688
- relaxng() (pynlpl.formats.folia.Entity class method), 558 relaxng() (pynlpl.formats.folia.Entry class method), 146
- relaxng() (pynlpl.formats.folia.ErrorDetection class
- method), 970 relaxng() (pynlpl.formats.folia.Event class method), 159 relaxng() (pynlpl.formats.folia.Example class method),

172

- $relaxng()\ (pynlpl.formats.folia. Feature\ class\ method),\ 868$
- relaxng() (pynlpl.formats.folia.Figure class method), 185 relaxng() (pynlpl.formats.folia.Gap class method), 196
- relaxing() (pynipi.formats.folia.Gap class method), 190 relaxing() (pynipi.formats.folia.Head class method), 209
- relaxng() (pynlpl.formats.folia.Headspan class method),
- relaxng() (pynlpl.formats.folia.LangAnnotation class method), 456
- relaxng() (pynlpl.formats.folia.LemmaAnnotation class method), 467
- relaxng() (pynlpl.formats.folia.Linebreak class method),
- relaxng() (pynlpl.formats.folia.List class method), 235
- relaxng() (pynlpl.formats.folia.ListItem class method),
- relaxng() (pynlpl.formats.folia.Metric class method), 1048
- relaxng() (pynlpl.formats.folia.New class method), 981
- relaxng() (pynlpl.formats.folia.Note class method), 261
- relaxng() (pynlpl.formats.folia.Observation class method), 570
- relaxng() (pynlpl.formats.folia.ObservationLayer class method), 699
- relaxng() (pynlpl.formats.folia.Original class method), 992
- relaxng() (pynlpl.formats.folia.Paragraph class method), 274
- relaxng() (pynlpl.formats.folia.Part class method), 287
- relaxng() (pynlpl.formats.folia.PhonContent class method), 510
- relaxng() (pynlpl.formats.folia.PosAnnotation class method), 445
- relaxng() (pynlpl.formats.folia.Predicate class method), 582
- $relaxng()\ (pynlpl.formats.folia.Quote\ class\ method),\ 300$
- relaxng() (pynlpl.formats.folia.Reference class method), 313
- relaxng() (pynlpl.formats.folia.Row class method), 326 relaxng() (pynlpl.formats.folia.SemanticRole clas
- method), 628 relaxng() (pynlpl.formats.folia.SemanticRolesLayer class
- relaxng() (pynlpl.formats.folia.SenseAnnotation class

method), 746

method), 478 relaxng() (pynlpl.formats.folia.Sentence class method), relaxng() (pynlpl.formats.folia.Sentiment class method), (pynlpl.formats.folia.SentimentLayer relaxng() class method), 711 relaxng() (pynlpl.formats.folia.Statement class method), (pynlpl.formats.folia.StatementLayer relaxng() class method), 723 (pynlpl.formats.folia.SubjectivityAnnotation relaxng() class method), 489 relaxng() (pynlpl.formats.folia.Suggestion class method), relaxng() (pynlpl.formats.folia.SynsetFeature class method), 879 relaxng() (pynlpl.formats.folia.SyntacticUnit class method), 617 (pynlpl.formats.folia.SyntaxLayer relaxng() class method), 735 relaxng() (pynlpl.formats.folia.Table class method), 354 relaxng() (pynlpl.formats.folia.TableHead class method), relaxng() (pynlpl.formats.folia.Term class method), 367 relaxng() (pynlpl.formats.folia.Text class method), 393 relaxng() (pynlpl.formats.folia.TextContent class method), 500 (pynlpl.formats.folia.TextMarkupCorrection relaxng() class method), 845 relaxng() (pynlpl.formats.folia.TextMarkupError class method), 856 (pynlpl.formats.folia.TextMarkupGap relaxng() class method), 813 relaxng() (pynlpl.formats.folia.TextMarkupString class method), 824 relaxng() (pynlpl.formats.folia.TextMarkupStyle class method), 835 (pynlpl.formats.folia.TimeSegment relaxng() class method), 640 relaxng() (pynlpl.formats.folia.TimingLayer class method), 758 relaxng() (pynlpl.formats.folia.Whitespace class method), 406 relaxng() (pynlpl.formats.folia.Word class method), 421 remove() (pynlpl.datatypes.PatternSet method), 5 remove() (pynlpl.formats.folia.AbstractAnnotationLayer method), 80

remove() (pynlpl.formats.folia.AbstractElement method),

remove() (pynlpl.formats.folia.AbstractSpanAnnotation

remove() (pynlpl.formats.folia.AbstractStructureElement

method), 57

method), 44

remove() (pynlpl.formats.folia.AbstractTextMarkup method), 91 remove() (pynlpl.formats.folia.AbstractTokenAnnotation method), 69 remove() (pynlpl.formats.folia.ActorFeature method), remove() (pynlpl.formats.folia.Alignment method), 1014 remove() (pynlpl.formats.folia.AlignReference method), 1025 remove() (pynlpl.formats.folia.Alternative method), 925 remove() (pynlpl.formats.folia.AlternativeLayers method), 936 (pynlpl.formats.folia.BegindatetimeFeature remove() method), 901 remove() (pynlpl.formats.folia.Cell method), 107 remove() (pynlpl.formats.folia.Chunk method), 523 remove() (pynlpl.formats.folia.ChunkingLayer method), 652 remove() (pynlpl.formats.folia.CoreferenceChain method), 535 (pynlpl.formats.folia.CoreferenceLayer remove() method), 664 remove() (pynlpl.formats.folia.CoreferenceLink method), remove() (pynlpl.formats.folia.Correction method), 948 remove() (pynlpl.formats.folia.Current method), 959 remove() (pynlpl.formats.folia.Definition method), 120 (pynlpl.formats.folia.DependenciesLayer remove() method), 676 remove() (pynlpl.formats.folia.Dependency method), 547 (pynlpl.formats.folia.DependencyDependent remove() method), 782 remove() (pynlpl.formats.folia.Description method), 1037 remove() (pynlpl.formats.folia.Division method), 133 (pynlpl.formats.folia.DomainAnnotation remove() method), 434 remove() (pynlpl.formats.folia.EnddatetimeFeature method), 912 remove() (pynlpl.formats.folia.EntitiesLayer method), remove() (pynlpl.formats.folia.Entity method), 558 remove() (pynlpl.formats.folia.Entry method), 146 remove() (pynlpl.formats.folia.ErrorDetection method), 970 remove() (pynlpl.formats.folia.Event method), 159 remove() (pynlpl.formats.folia.Example method), 172 remove() (pynlpl.formats.folia.Feature method), 868 remove() (pynlpl.formats.folia.Figure method), 185 remove() (pynlpl.formats.folia.Gap method), 196 remove() (pynlpl.formats.folia.Head method), 209 remove() (pynlpl.formats.folia.Headspan method), 793 remove() (pynlpl.formats.folia.LangAnnotation method),

1148 Index

remove()

456

(pynlpl.formats.folia.LemmaAnnotation

- method), 467 remove() (pynlpl.formats.folia.Linebreak method), 222 remove() (pynlpl.formats.folia.List method), 235 remove() (pynlpl.formats.folia.ListItem method), 248 remove() (pynlpl.formats.folia.Metric method), 1048 remove() (pynlpl.formats.folia.New method), 981 remove() (pynlpl.formats.folia.Note method), 261 remove() (pynlpl.formats.folia.Observation method), 570 remove() (pynlpl.formats.folia.ObservationLayer method), 699 remove() (pynlpl.formats.folia.Original method), 992 remove() (pynlpl.formats.folia.Paragraph method), 274 remove() (pynlpl.formats.folia.Part method), 287 remove() (pynlpl.formats.folia.PhonContent method), 510 remove() (pynlpl.formats.folia.PosAnnotation method), remove() (pynlpl.formats.folia.Predicate method), 582 remove() (pynlpl.formats.folia.Quote method), 300 remove() (pynlpl.formats.folia.Reference method), 313 remove() (pynlpl.formats.folia.Row method), 326 remove() (pynlpl.formats.folia.SemanticRole method), 628 (pynlpl.formats.folia.SemanticRolesLaver remove() method), 746 remove() (pynlpl.formats.folia.SenseAnnotation method), 478 remove() (pynlpl.formats.folia.Sentence method), 341 remove() (pynlpl.formats.folia.Sentiment method), 593 remove() (pynlpl.formats.folia.SentimentLayer method), remove() (pynlpl.formats.folia.Statement method), 605 remove() (pynlpl.formats.folia.StatementLayer method), (pynlpl.formats.folia.SubjectivityAnnotation remove() method), 489 remove() (pynlpl.formats.folia.Suggestion method), 1003 remove() (pynlpl.formats.folia.SynsetFeature method), 879 remove() (pynlpl.formats.folia.SyntacticUnit method), remove() (pynlpl.formats.folia.SyntaxLayer method), 735 remove() (pynlpl.formats.folia.Table method), 354 remove() (pynlpl.formats.folia.TableHead method), 380 remove() (pynlpl.formats.folia.Term method), 367 remove() (pynlpl.formats.folia.Text method), 393 remove() (pynlpl.formats.folia.TextContent method), 500 remove() (pynlpl.formats.folia.TextMarkupCorrection method), 845 (pynlpl.formats.folia.TextMarkupError remove() method), 856 remove() (pynlpl.formats.folia.TextMarkupGap method), 813
- remove() (pynlpl.formats.folia.TextMarkupString method), 824
- remove() (pynlpl.formats.folia.TextMarkupStyle method), 835
- remove() (pynlpl.formats.folia.TimeSegment method), 640
- remove() (pynlpl.formats.folia.TimingLayer method), 758
- remove() (pynlpl.formats.folia.Whitespace method), 406 remove() (pynlpl.formats.folia.Word method), 421
- replace() (pynlpl.formats.folia.AbstractAnnotationLayer method), 81
- $replace() \ (pynlpl.formats.folia. Abstract Element \ method), \\ 31$
- replace() (pynlpl.formats.folia.AbstractSpanAnnotation method), 58
- replace() (pynlpl.formats.folia.AbstractStructureElement method), 44
- replace() (pynlpl.formats.folia.AbstractTextMarkup method), 91
- $\begin{tabular}{ll} replace() & (pynlpl.formats.folia. Abstract Token Annotation \\ & method), 69 \end{tabular}$
- replace() (pynlpl.formats.folia.ActorFeature method), 890
- replace() (pynlpl.formats.folia.Alignment method), 1014
- replace() (pynlpl.formats.folia.AlignReference method), 1025
- replace() (pynlpl.formats.folia.Alternative method), 925 replace() (pynlpl.formats.folia.AlternativeLayers method), 937
- replace() (pynlpl.formats.folia.BegindatetimeFeature method), 901
- replace() (pynlpl.formats.folia.Cell method), 107
- replace() (pynlpl.formats.folia.Chunk method), 524
- replace() (pynlpl.formats.folia.ChunkingLayer method), 652
- replace() (pynlpl.formats.folia.CoreferenceChain method), 535
- replace() (pynlpl.formats.folia.CoreferenceLayer method), 664
- replace() (pynlpl.formats.folia.CoreferenceLink method), 770
- replace() (pynlpl.formats.folia.Correction method), 948
- replace() (pynlpl.formats.folia.Current method), 959
- replace() (pynlpl.formats.folia.Definition method), 120
- replace() (pynlpl.formats.folia.DependenciesLayer method), 676
- replace() (pynlpl.formats.folia.Dependency method), 547 replace() (pynlpl.formats.folia.DependencyDependent method), 782
- replace() (pynlpl.formats.folia.Description method), 1037 replace() (pynlpl.formats.folia.Division method), 133
- replace() (pynlpl.formats.folia.DomainAnnotation method), 434

(pynlpl.formats.folia.EnddatetimeFeature replace() (pynlpl.formats.folia.SynsetFeature method), replace() method), 912 replace() (pynlpl.formats.folia.EntitiesLayer method), replace() (pynlpl.formats.folia.SyntacticUnit method), replace() (pynlpl.formats.folia.Entity method), 558 replace() (pynlpl.formats.folia.SyntaxLayer method), 735 replace() (pynlpl.formats.folia.Entry method), 146 replace() (pynlpl.formats.folia.Table method), 354 replace() (pynlpl.formats.folia.ErrorDetection method). replace() (pynlpl.formats.folia.TableHead method), 380 replace() (pynlpl.formats.folia.Term method), 367 replace() (pynlpl.formats.folia.Event method), 159 replace() (pynlpl.formats.folia.Text method), 393 replace() (pynlpl.formats.folia.Example method), 172 replace() (pynlpl.formats.folia.TextContent method), 500 replace() (pynlpl.formats.folia.Feature method), 868 replace() (pynlpl.formats.folia.TextMarkupCorrection replace() (pynlpl.formats.folia.Figure method), 185 method), 845 (pynlpl.formats.folia.TextMarkupError replace() (pynlpl.formats.folia.Gap method), 196 replace() replace() (pynlpl.formats.folia.Head method), 209 method), 856 replace() (pynlpl.formats.folia.Headspan method), 793 replace() (pynlpl.formats.folia.TextMarkupGap method), replace() (pynlpl.formats.folia.LangAnnotation method), 813 456 replace() (pynlpl.formats.folia.TextMarkupString (pynlpl.formats.folia.LemmaAnnotation method), 824 replace() method), 467 replace() (pynlpl.formats.folia.TextMarkupStyle method), 835 replace() (pynlpl.formats.folia.Linebreak method), 222 replace() (pynlpl.formats.folia.TimeSegment method), replace() (pynlpl.formats.folia.List method), 235 replace() (pynlpl.formats.folia.ListItem method), 248 replace() (pynlpl.formats.folia.Metric method), 1048 replace() (pynlpl.formats.folia.TimingLayer method), 758 replace() (pynlpl.formats.folia.New method), 981 replace() (pynlpl.formats.folia.Whitespace method), 406 replace() (pynlpl.formats.folia.Note method), 261 replace() (pynlpl.formats.folia.Word method), 421 replace() (pynlpl.formats.folia.Observation method), 570 REQUIRED ATTRIBS (pynlpl.formats.folia.AbstractAnnotationLayer replace() (pynlpl.formats.folia.ObservationLayer attribute), 75 method), 700 REQUIRED\_ATTRIBS (pynlpl.formats.folia.AbstractElement replace() (pynlpl.formats.folia.Original method), 992 attribute), 26 replace() (pynlpl.formats.folia.Paragraph method), 274 REQUIRED\_ATTRIBS (pynlpl.formats.folia.AbstractSpanAnnotation replace() (pynlpl.formats.folia.Part method), 287 attribute), 52 replace() (pynlpl.formats.folia.PhonContent method), 510 REQUIRED\_ATTRIBS (pynlpl.formats.folia.AbstractStructureElement replace() (pynlpl.formats.folia.PosAnnotation method), attribute), 37 REQUIRED\_ATTRIBS (pynlpl.formats.folia.AbstractTextMarkup replace() (pynlpl.formats.folia.Predicate method), 582 attribute), 86 replace() (pynlpl.formats.folia.Quote method), 300 REQUIRED ATTRIBS (pynlpl.formats.folia.AbstractTokenAnnotation replace() (pynlpl.formats.folia.Reference method), 313 attribute), 63 replace() (pynlpl.formats.folia.Row method), 326 REQUIRED\_ATTRIBS (pynlpl.formats.folia.ActorFeature replace() (pynlpl.formats.folia.SemanticRole method), attribute), 885 628 REQUIRED\_ATTRIBS (pynlpl.formats.folia.Alignment replace() (pynlpl.formats.folia.SemanticRolesLayer attribute), 1009 REQUIRED ATTRIBS (pynlpl.formats.folia.AlignReference method), 747 replace() (pynlpl.formats.folia.SenseAnnotation method), attribute), 1020 478 REQUIRED\_ATTRIBS (pynlpl.formats.folia.Alternative replace() (pynlpl.formats.folia.Sentence method), 341 attribute), 919 replace() (pynlpl.formats.folia.Sentiment method), 593 REQUIRED\_ATTRIBS (pynlpl.formats.folia.AlternativeLayers replace() (pynlpl.formats.folia.SentimentLayer method), attribute), 931 REQUIRED\_ATTRIBS (pynlpl.formats.folia.BegindatetimeFeature replace() (pynlpl.formats.folia.Statement method), 605 attribute), 896 replace() (pynlpl.formats.folia.StatementLayer method), REQUIRED\_ATTRIBS (pynlpl.formats.folia.Cell attribute), 100 REQUIRED\_ATTRIBS (pynlpl.formats.folia.Chunk at-(pynlpl.formats.folia.SubjectivityAnnotation replace() method), 489 tribute), 518 replace() (pynlpl.formats.folia.Suggestion method), 1003 REQUIRED ATTRIBS (pynlpl.formats.folia.ChunkingLayer

auditura) 646
attribute), 646 attribute), 215 REQUIRED_ATTRIBS (pynlpl.formats.folia.CoreferenceClrique) (pynlpl.formats.folia.List at-
attribute), 529 tribute), 228
REQUIRED_ATTRIBS (pynlpl.formats.folia.CoreferenceLaREQUIRED_ATTRIBS (pynlpl.formats.folia.ListItem atattribute), 658 tribute), 241
REQUIRED_ATTRIBS (pynlpl.formats.folia.CoreferenceLiREQUIRED_ATTRIBS (pynlpl.formats.folia.Metric at-
attribute), 764 tribute), 1042
REQUIRED_ATTRIBS (pynlpl.formats.folia.Correction REQUIRED_ATTRIBS (pynlpl.formats.folia.New
attribute), 944 attribute), 976 REQUIRED_ATTRIBS (pynlpl.formats.folia.Current at- REQUIRED_ATTRIBS (pynlpl.formats.folia.Note
tribute), 953 attribute), 254
REQUIRED_ATTRIBS (pynlpl.formats.folia.Definition REQUIRED_ATTRIBS (pynlpl.formats.folia.Observation attribute), 113
REQUIRED_ATTRIBS (pynlpl.formats.folia.Dependencies REQUIRED_ATTRIBS (pynlpl.formats.folia.ObservationLayer
attribute), 670 attribute), 694
REQUIRED_ATTRIBS (pynlpl.formats.folia.Dependency REQUIRED_ATTRIBS (pynlpl.formats.folia.Original at-
attribute), 541 tribute), 987
REQUIRED_ATTRIBS (pynlpl.formats.folia.DependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDepende
attribute), 776 attribute), 267
REQUIRED_ATTRIBS (pynlpl.formats.folia.Description REQUIRED_ATTRIBS (pynlpl.formats.folia.Part at-attribute), 1031 tribute), 280
REQUIRED_ATTRIBS (pynlpl.formats.folia.Division at-
tribute), 126 attribute), 506
REQUIRED_ATTRIBS (pynlpl.formats.folia.DomainAnnotREQUIRED_ATTRIBS (pynlpl.formats.folia.PosAnnotation
attribute), 428 attribute), 440
REQUIRED_ATTRIBS (pynlpl.formats.folia.EnddatetimeF&EQUIRED_ATTRIBS (pynlpl.formats.folia.Predicate
attribute), 907 attribute), 576
REQUIRED_ATTRIBS (pynlpl.formats.folia.EntitiesLayer REQUIRED_ATTRIBS (pynlpl.formats.folia.Quote at-
attribute), 682 tribute), 293
REQUIRED_ATTRIBS (pynlpl.formats.folia.Entity at- REQUIRED_ATTRIBS (pynlpl.formats.folia.Reference
tribute), 553  attribute), 306  PEOLUPED ATTRIBS (pynlal formats folio Entry, et PEOLUPED ATTRIBS (pynlal formats folio Poyr
REQUIRED_ATTRIBS (pynlpl.formats.folia.Entry at REQUIRED_ATTRIBS (pynlpl.formats.folia.Row attribute), 319 (pynlpl.formats.folia.Row
REQUIRED_ATTRIBS (pynlpl.formats.folia.ErrorDetectiorREQUIRED_ATTRIBS (pynlpl.formats.folia.SemanticRole
attribute), 964 attribute), 623
REQUIRED_ATTRIBS (pynlpl.formats.folia.Event at REQUIRED_ATTRIBS (pynlpl.formats.folia.SemanticRolesLayer
tribute), 152 attribute), 741
REQUIRED_ATTRIBS (pynlpl.formats.folia.Example REQUIRED_ATTRIBS (pynlpl.formats.folia.SenseAnnotation
attribute), 165 attribute), 473
REQUIRED_ATTRIBS (pynlpl.formats.folia.Feature at- REQUIRED_ATTRIBS (pynlpl.formats.folia.Sentence
tribute), 862 attribute), 333
REQUIRED_ATTRIBS (pynlpl.formats.folia.Figure at- REQUIRED_ATTRIBS (pynlpl.formats.folia.Sentiment
tribute), 178 attribute), 588 REQUIRED_ATTRIBS (pynlpl.formats.folia.Gap at- REQUIRED_ATTRIBS (pynlpl.formats.folia.SentimentLayer
REQUIRED_ATTRIBS (pynlpl.formats.folia.Gap at REQUIRED_ATTRIBS (pynlpl.formats.folia.SentimentLayer tribute), 191 attribute), 705
REQUIRED_ATTRIBS (pynlpl.formats.folia.Head at- REQUIRED_ATTRIBS (pynlpl.formats.folia.Statement
tribute), 202 attribute), 599
REQUIRED_ATTRIBS (pynlpl.formats.folia.Headspan REQUIRED_ATTRIBS (pynlpl.formats.folia.StatementLayer
attribute), 788 attribute), 717
$REQUIRED\_ATTRIBS\ (pynlpl. formats. folia. Lang Annotation Part College Coll$
attribute), 451 attribute), 484
REQUIRED_ATTRIBS (pynlpl.formats.folia.LemmaAnnotaREQUIRED_ATTRIBS (pynlpl.formats.folia.Suggestion
attribute), 462 attribute), 998
REQUIRED_ATTRIBS (pynlpl.formats.folia.Linebreak REQUIRED_ATTRIBS (pynlpl.formats.folia.SynsetFeature

attribute), 874 attribute), 931
REQUIRED_ATTRIBS (pynlpl.formats.folia.SyntacticUnit REQUIRED_DATA (pynlpl.formats.folia.BegindatetimeFeature
attribute), 611 attribute), 896
REQUIRED_ATTRIBS (pynlpl.formats.folia.SyntaxLayer REQUIRED_DATA (pynlpl.formats.folia.Cell attribute),
attribute), 729
REQUIRED_ATTRIBS (pynlpl.formats.folia.Table at- REQUIRED_DATA (pynlpl.formats.folia.Chunk at-
tribute), 347 tribute), 518
REQUIRED_ATTRIBS (pynlpl.formats.folia.TableHead REQUIRED_DATA (pynlpl.formats.folia.ChunkingLayer
attribute), 373 attribute), 646
REQUIRED_ATTRIBS (pynlpl.formats.folia.Term at REQUIRED_DATA (pynlpl.formats.folia.CoreferenceChain
tribute), 360 attribute), 530
REQUIRED_ATTRIBS (pynlpl.formats.folia.Text REQUIRED_DATA (pynlpl.formats.folia.CoreferenceLayer
attribute), 386 attribute), 658
REQUIRED_ATTRIBS (pynlpl.formats.folia.TextContent REQUIRED_DATA (pynlpl.formats.folia.CoreferenceLink
attribute), 495 attribute), 764
REQUIRED_ATTRIBS (pynlpl.formats.folia.TextMarkupCorrection at-
attribute), 840 tribute), 944
REQUIRED_ATTRIBS (pynlpl.formats.folia.TextMarkupEnReQUIRED_DATA (pynlpl.formats.folia.Current at-
attribute), 851 tribute), 953
REQUIRED_ATTRIBS (pynlpl.formats.folia.TextMarkupGREQUIRED_DATA (pynlpl.formats.folia.Definition at-
attribute), 808 tribute), 113
REQUIRED_ATTRIBS (pynlpl.formats.folia.TextMarkupStREQUIRED_DATA (pynlpl.formats.folia.DependenciesLayer
attribute), 819 attribute), 670
REQUIRED_ATTRIBS (pynlpl.formats.folia.TextMarkupStREQUIRED_DATA (pynlpl.formats.folia.Dependency
attribute), 830 attribute), 541
REQUIRED_ATTRIBS (pynlpl.formats.folia.TimeSegmentREQUIRED_DATA (pynlpl.formats.folia.DependencyDependent
attribute), 634 attribute), 776
REQUIRED_ATTRIBS (pynlpl.formats.folia.TimingLayer REQUIRED_DATA (pynlpl.formats.folia.Description at-
attribute), 752 tribute), 1031
REQUIRED_ATTRIBS (pynlpl.formats.folia.Whitespace REQUIRED_DATA (pynlpl.formats.folia.Division
attribute), 399 attribute), 126
REQUIRED_ATTRIBS (pynlpl.formats.folia.Word at- REQUIRED_DATA (pynlpl.formats.folia.DomainAnnotation
tribute), 413 attribute), 429
REQUIRED_DATA (pynlpl.formats.folia.AbstractAnnotationREQUIRED_DATA (pynlpl.formats.folia.EnddatetimeFeature
attribute), 75 attribute), 907
REQUIRED_DATA (pynlpl.formats.folia.AbstractElement REQUIRED_DATA (pynlpl.formats.folia.EntitiesLayer
attribute), 26 attribute), 682
REQUIRED_DATA (pynlpl.formats.folia.AbstractSpanAnn&EQuIRED_DATA (pynlpl.formats.folia.Entity at-
attribute), 52 tribute), 553
REQUIRED_DATA (pynlpl.formats.folia.AbstractStructureREQENTRED_DATA (pynlpl.formats.folia.Entry at-
attribute), 37 tribute), 139
REQUIRED_DATA (pynlpl.formats.folia.AbstractTextMarkREQUIRED_DATA (pynlpl.formats.folia.ErrorDetection
attribute), 86 attribute), 965
REQUIRED_DATA (pynlpl.formats.folia.AbstractTokenAniREQUIRED_DATA (pynlpl.formats.folia.Event at-
attribute), 63 tribute), 152
REQUIRED_DATA (pynlpl.formats.folia.ActorFeature REQUIRED_DATA (pynlpl.formats.folia.Example at-
attribute), 885 tribute), 165
REQUIRED_DATA (pynlpl.formats.folia.Alignment at- REQUIRED_DATA (pynlpl.formats.folia.Feature at-
tribute), 1009 tribute), 862
REQUIRED_DATA (pynlpl.formats.folia.AlignReference REQUIRED_DATA (pynlpl.formats.folia.Figure at-
attribute), 1020 tribute), 178
REQUIRED_DATA (pynlpl.formats.folia.Alternative at- REQUIRED_DATA (pynlpl.formats.folia.Gap attribute),
tribute), 919 191
REQUIRED, DATA (nynlpl formats folia Alternativel avers REQUIRED, DATA (nynlpl formats folia Head attribute)

202 tribute), 599	
REQUIRED_DATA (pynlpl.formats.folia.Headspan at REQUIRED_DATA (pynlpl.formats.folia.Statemen	ntLayer
tribute), 788 attribute), 717	
REQUIRED_DATA (pynlpl.formats.folia.LangAnnotation REQUIRED_DATA (pynlpl.formats.folia.Subjective attribute), 451 attribute), 484	vityAnnotation
attribute), 451 attribute), 484 REQUIRED_DATA (pynlpl.formats.folia.LemmaAnnotatioiREQUIRED_DATA (pynlpl.formats.folia.Suggesti	ion at-
attribute), 462 tribute), 998	ion at
REQUIRED_DATA (pynlpl.formats.folia.Linebreak at- REQUIRED_DATA (pynlpl.formats.folia.SynsetF	Feature
tribute), 215 attribute), 874	
REQUIRED_DATA (pynlpl.formats.folia.List attribute), REQUIRED_DATA (pynlpl.formats.folia.Syntact attribute), 611	ticUnit
REQUIRED_DATA (pynlpl.formats.folia.ListItem REQUIRED_DATA (pynlpl.formats.folia.Syntax attribute), 241	xLayer
REQUIRED_DATA (pynlpl.formats.folia.Metric at- REQUIRED_DATA (pynlpl.formats.folia.Table	e at-
tribute), 1043 tribute), 347	
REQUIRED_DATA (pynlpl.formats.folia.New attribute), REQUIRED_DATA (pynlpl.formats.folia.TableHe 976 tribute), 373	ead at-
976 tribute), 373 REQUIRED_DATA (pynlpl.formats.folia.Note attribute), REQUIRED_DATA (pynlpl.formats.folia.Term attr	ribute)
254 360	iloute),
REQUIRED_DATA (pynlpl.formats.folia.Observation attribute), 564  REQUIRED_DATA (pynlpl.formats.folia.Text attribute), 386	ribute),
REQUIRED_DATA (pynlpl.formats.folia.ObservationLayerREQUIRED_DATA (pynlpl.formats.folia.TextCont	tent at-
attribute), 694 tribute), 495	
REQUIRED_DATA (pynlpl.formats.folia.Original REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 987 attribute), 840	kupCorrection
REQUIRED_DATA (pynlpl.formats.folia.Paragraph attribute), 267 REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 851	kupError
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), REQUIRED_DATA (pynlpl.formats.folia.TextMark	kupGap
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280 REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 808 REQUIRED_DATA (pynlpl.formats.folia.PhonContent REQUIRED_DATA (pynlpl.formats.folia.TextMark Required	
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 808  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 819	kupString
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 808  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 819  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 819  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 819  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 819	kupString
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280 attribute), 808  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506 REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440 attribute), 830  REQUIRED_DATA (pynlpl.formats.folia.Predicate at-REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830	kupString kupStyle
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830  REQUIRED_DATA (pynlpl.formats.folia.TimeSetribute), 576  REQUIRED_DATA (pynlpl.formats.folia.TimeSetribute), 634	kupString kupStyle egment
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 819  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830	kupString kupStyle egment
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830	kupString kupStyle egment gLayer
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 819  REQUIRED_DATA (pynlpl.formats.folia.TextMark attribute), 830	kupString kupStyle egment gLayer
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 399  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), REQUIRED_DATA (pynlpl.formats.folia.Whitespartribute), 399  REQUIRED_DATA (pynlpl.formats.folia.Word	kupString kupStyle egment gLayer vace at-
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 399  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), 319  REQUIRED_DATA (pynlpl.formats.folia.Whitespartibute), 319  REQUIRED_DATA (pynlpl.formats.folia.Whitespartibute), 319	kupString kupStyle egment gLayer ace at-
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 399  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), REQUIRED_DATA (pynlpl.formats.folia.Whitespartribute), 399  REQUIRED_DATA (pynlpl.formats.folia.Word	kupString kupStyle egment gLayer ace at-
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), 319  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRoles attribute), 623  REQUIRED_DATA (pynlpl.formats.giza.MultiWordAlig	kupString kupStyle egment gLayer ace at- l at- gnment
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reserence attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), 319  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 741  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation reset()  REQUIRED_DATA (pynlpl.formats.giza.WordAlignment method), 1053  REQUIRED_DATA (pynlpl.formats.giza.WordAlignment method), 1053  REQUIRED_DATA (pynlpl.formats.giza.WordAlignment method), 1053  REQUIRED_DATA (pynlpl.formats.giza.WordAlignment method), 1053	kupString kupStyle egment gLayer vace at- I at- gnment gnment
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), 319  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 741  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation reset() attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 489  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 480  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.Tex	kupString kupStyle egment gLayer vace at- l at- gnment gnment ethod),
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), 319  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRoles attribute), 741  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333	kupString kupStyle egment gLayer eace at- l at- gnment gnment ethod), ethod),
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), 319  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 741  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation reset() attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 489  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 480  REQUIRED_DATA (pynlpl.formats.folia.TextMart attribute), 481  REQUIRED_DATA (pynlpl.formats.folia.Tex	kupString kupStyle egment gLayer eace at- at- gnment gnment ethod), ethod),
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRole attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRoles attribute), 741  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation reset() (pynlpl.formats.giza.MultiWordAlig method), 1053  REQUIRED_DATA (pynlpl.formats.folia.Sentiment attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentiment attribute), 588  REQUIRED_DATA (pynlpl.formats.folia.SentimentLayer	kupString kupStyle egment gLayer pace at- at- gnment gnment ethod), ethod), Markup
REQUIRED_DATA (pynlpl.formats.folia.Part attribute), 280  REQUIRED_DATA (pynlpl.formats.folia.PhonContent attribute), 506  REQUIRED_DATA (pynlpl.formats.folia.PosAnnotation attribute), 440  REQUIRED_DATA (pynlpl.formats.folia.Predicate attribute), 576  REQUIRED_DATA (pynlpl.formats.folia.Quote attribute), 293  REQUIRED_DATA (pynlpl.formats.folia.Reference attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Research attribute), 306  REQUIRED_DATA (pynlpl.formats.folia.Row attribute), 319  REQUIRED_DATA (pynlpl.formats.folia.SemanticRoles attribute), 623  REQUIRED_DATA (pynlpl.formats.folia.SemanticRoles attribute), 741  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.SenseAnnotation reset() (pynlpl.formats.giza.WordAlignment method), 1053  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 358  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 368  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 473  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 333  REQUIRED_DATA (pynlpl.formats.folia.Sentence attribute), 358	kupString kupStyle egment gLayer vace at- l at- gnment gnment ethod), ethod), f9 Markup , 1014

1026	resolveword() (pynlpl.formats.folia.Dependency
resolve() (pynlpl.formats.folia.Linebreak method), 222	method), 547
resolve() (pynlpl.formats.folia.Reference method), 313 resolve() (pynlpl.formats.folia.TextMarkupCorrection	resolveword() (pynlpl.formats.folia.DependencyDependent method), 782
method), 846	resolveword() (pynlpl.formats.folia.Description method),
resolve() (pynlpl.formats.folia.TextMarkupError	1037
method), 856	resolveword() (pynlpl.formats.folia.Division method),
resolve() (pynlpl.formats.folia.TextMarkupGap method),	133
814	$resolveword () \\ \hspace{0.5cm} (pynlpl.formats.folia. Domain Annotation$
resolve() (pynlpl.formats.folia.TextMarkupString	method), 434
method), 824	resolveword() (pynlpl.formats.folia.EnddatetimeFeature
resolve() (pynlpl.formats.folia.TextMarkupStyle	method), 912
method), 835	resolveword() (pynlpl.formats.folia.EntitiesLayer
resolveword() (pynlpl.formats.folia.AbstractAnnotationLay	
method), 81	resolveword() (pynlpl.formats.folia.Entity method), 559 resolveword() (pynlpl.formats.folia.Entry method), 146
resolveword() (pynlpl.formats.folia.AbstractElement method), 31	resolveword() (pynlpl.formats.folia.Entry method), 140 resolveword() (pynlpl.formats.folia.ErrorDetection
resolveword() (pynlpl.formats.folia.AbstractSpanAnnotatio	
method), 58	resolveword() (pynlpl.formats.folia.Event method), 159
resolveword() (pynlpl.formats.folia.AbstractStructureElemo	
method), 44	172
resolveword() (pynlpl.formats.folia.AbstractTextMarkup	resolveword() (pynlpl.formats.folia.Feature method), 868
method), 92	resolveword() (pynlpl.formats.folia.Figure method), 185
$resolveword () \ (pynlpl. formats. folia. Abstract Token Annotation (pynlpl. formats. folia.) Abstract Token Annotation (pynlpl. formats. formats. folia.) Abstract Token Annotation (pynlpl. formats. formats. formats. formats. formats. formats. formats. formats. formats. forma$	
method), 69	resolveword() (pynlpl.formats.folia.Head method), 209
resolveword() (pynlpl.formats.folia.ActorFeature method), 890	resolveword() (pynlpl.formats.folia.Headspan method), 794
resolveword() (pynlpl.formats.folia.Alignment method),	resolveword() (pynlpl.formats.folia.LangAnnotation
1014	method), 456
resolveword() (pynlpl.formats.folia.AlignReference	resolveword() (pynlpl.formats.folia.LemmaAnnotation
method), 1026	method), 467
resolveword() (pynlpl.formats.folia.Alternative method),	resolveword() (pynlpl.formats.folia.Linebreak method),
926	222
resolveword() (pynlpl.formats.folia.AlternativeLayers	resolveword() (pynlpl.formats.folia.List method), 235
method), 937	resolveword() (pynlpl.formats.folia.ListItem method),
resolveword() (pynlpl.formats.folia.BegindatetimeFeature	248
method), 901	resolveword() (pynlpl.formats.folia.Metric method), 1048
resolveword() (pynlpl.formats.folia.Cell method), 107 resolveword() (pynlpl.formats.folia.Chunk method), 524	resolveword() (pynlpl.formats.folia.New method), 981
resolveword() (pympi.formats.folia.Chunk method), 324 resolveword() (pynlpl.formats.folia.ChunkingLayer	resolveword() (pynlpl.formats.folia.Note method), 261 resolveword() (pynlpl.formats.folia.Observation method),
method), 653	570
resolveword() (pynlpl.formats.folia.CoreferenceChain	resolveword() (pynlpl.formats.folia.ObservationLayer
method), 535	method), 700
resolveword() (pynlpl.formats.folia.CoreferenceLayer method), 664	resolveword() (pynlpl.formats.folia.Original method),
resolveword() (pynlpl.formats.folia.CoreferenceLink	resolveword() (pynlpl.formats.folia.Paragraph method),
method), 770	274
resolveword() (pynlpl.formats.folia.Correction method),	resolveword() (pynlpl.formats.folia.Part method), 287
949	resolveword() (pynlpl.formats.folia.PhonContent
resolveword() (pynlpl.formats.folia.Current method), 959	method), 510
resolveword() (pynlpl.formats.folia.Definition method), 120	resolveword() (pynlpl.formats.folia.PosAnnotation method), 445
resolveword() (pynlpl.formats.folia.DependenciesLayer	resolveword() (pynlpl.formats.folia.Predicate method),
method), 676	582

- resolveword() (pynlpl.formats.folia.Quote method), 300 resolveword() (pynlpl.formats.folia.Reference method), resolveword() (pynlpl.formats.folia.Row method), 326 (pynlpl.formats.folia.SemanticRole resolveword() method), 629 resolveword() (pynlpl.formats.folia.SemanticRolesLayer method), 747 resolveword() (pynlpl.formats.folia.SenseAnnotation method), 478 resolveword() (pynlpl.formats.folia.Sentence method), 341 resolveword() (pynlpl.formats.folia.Sentiment method), 594 resolveword() (pynlpl.formats.folia.SentimentLayer method), 711 resolveword() (pynlpl.formats.folia.Statement method), 605 resolveword() method), 723 resolveword() (pynlpl.formats.folia.SubjectivityAnnotation\_rightcontext() method), 489 resolveword() (pynlpl.formats.folia.Suggestion method), rightcontext() (pynlpl.formats.folia.BegindatetimeFeature 1003 resolveword() (pynlpl.formats.folia.SynsetFeature method), 879 resolveword() (pynlpl.formats.folia.SyntacticUnit method), 617 (pynlpl.formats.folia.SyntaxLayer resolveword() method), 735 resolveword() (pynlpl.formats.folia.Table method), 354 resolveword() (pynlpl.formats.folia.TableHead method), resolveword() (pynlpl.formats.folia.Term method), 367 resolveword() (pynlpl.formats.folia.Text method), 393 resolveword() (pynlpl.formats.folia.TextContent method), resolveword() (pynlpl.formats.folia.TextMarkupCorrection rightcontext() (pynlpl.formats.folia.Definition method), method), 846 resolveword() (pynlpl.formats.folia.TextMarkupError resolveword() (pynlpl.formats.folia.TextMarkupGap method), 814 resolveword() (pynlpl.formats.folia.TextMarkupString method), 824 (pynlpl.formats.folia.TextMarkupStyle resolveword() method), 835 resolveword() (pynlpl.formats.folia.TimeSegment method), 640 (pynlpl.formats.folia.TimingLayer resolveword() method), 758 resolveword() (pynlpl.formats.folia.Whitespace method), resolveword() (pynlpl.formats.folia.Word method), 421 rightcontext()
- rightcontext() (pynlpl.formats.folia.AbstractAnnotationLayer method), 81 rightcontext() (pynlpl.formats.folia.AbstractElement method), 31 rightcontext() (pynlpl.formats.folia.AbstractSpanAnnotation method), 58 rightcontext() (pvnlpl.formats.folia.AbstractStructureElement method), 44 rightcontext() (pynlpl.formats.folia.AbstractTextMarkup method), 92 rightcontext() (pynlpl.formats.folia.AbstractTokenAnnotation method), 69 (pynlpl.formats.folia.ActorFeature rightcontext() method), 890 rightcontext() (pynlpl.formats.folia.Alignment method), 1014 rightcontext() (pynlpl.formats.folia.AlignReference method), 1026 (pynlpl.formats.folia.StatementLayer rightcontext() (pynlpl.formats.folia.Alternative method), (pynlpl.formats.folia.AlternativeLayers method), 937 method), 901 rightcontext() (pynlpl.formats.folia.Cell method), 107 rightcontext() (pynlpl.formats.folia.Chunk method), 524 rightcontext() (pynlpl.formats.folia.ChunkingLayer method), 653 (pynlpl.formats.folia.CoreferenceChain rightcontext() method), 535 rightcontext() (pynlpl.formats.folia.CoreferenceLayer method), 664 (pynlpl.formats.folia.CoreferenceLink rightcontext() method), 770 rightcontext() (pynlpl.formats.folia.Correction method), rightcontext() (pynlpl.formats.folia.Current method), 959 120 rightcontext() (pynlpl.formats.folia.DependenciesLayer method), 676 rightcontext() (pynlpl.formats.folia.Dependency method), rightcontext() (pynlpl.formats.folia.DependencyDependent method), 782 rightcontext() (pynlpl.formats.folia.Description method), 1037 rightcontext() (pynlpl.formats.folia.Division method), 133 (pynlpl.formats.folia.DomainAnnotation rightcontext() method), 434 rightcontext() (pynlpl.formats.folia.EnddatetimeFeature

method), 912

(pynlpl.formats.folia.EntitiesLayer

method), 688	594
rightcontext() (pynlpl.formats.folia.Entity method), 559	rightcontext() (pynlpl.formats.folia.SentimentLayer
rightcontext() (pynlpl.formats.folia.Entry method), 146	method), 711
rightcontext() (pynlpl.formats.folia.ErrorDetection	rightcontext() (pynlpl.formats.folia.Statement method),
method), 970	605
rightcontext() (pynlpl.formats.folia.Event method), 159	rightcontext() (pynlpl.formats.folia.StatementLayer
rightcontext() (pynlpl.formats.folia.Example method),	method), 723
172	rightcontext() (pynlpl.formats.folia.SubjectivityAnnotation
rightcontext() (pynlpl.formats.folia.Feature method), 868	method), 489
rightcontext() (pynlpl.formats.folia.Figure method), 185	
	rightcontext() (pynlpl.formats.folia.Suggestion method), 1003
rightcontext() (pynlpl.formats.folia.Gap method), 196	
rightcontext() (pynlpl.formats.folia.Head method), 209	rightcontext() (pynlpl.formats.folia.SynsetFeature
rightcontext() (pynlpl.formats.folia.Headspan method),	method), 879
794	rightcontext() (pynlpl.formats.folia.SyntacticUnit
rightcontext() (pynlpl.formats.folia.LangAnnotation	method), 617
method), 456	rightcontext() (pynlpl.formats.folia.SyntaxLayer
rightcontext() (pynlpl.formats.folia.LemmaAnnotation	method), 735
method), 467	rightcontext() (pynlpl.formats.folia.Table method), 354
rightcontext() (pynlpl.formats.folia.Linebreak method),	rightcontext() (pynlpl.formats.folia.TableHead method),
222	380
rightcontext() (pynlpl.formats.folia.List method), 235	rightcontext() (pynlpl.formats.folia.Term method), 367
rightcontext() (pynlpl.formats.folia.ListItem method),	rightcontext() (pynlpl.formats.folia.Text method), 393
248	rightcontext() (pynlpl.formats.folia.TextContent method),
rightcontext() (pynlpl.formats.folia.Metric method), 1048	501
rightcontext() (pynlpl.formats.folia.New method), 981	$right context () \ (pynlpl. formats. folia. Text Markup Correction$
rightcontext() (pynlpl.formats.folia.Note method), 261	method), 846
rightcontext() (pynlpl.formats.folia.Observation method),	rightcontext() (pynlpl.formats.folia.TextMarkupError
570	method), 856
rightcontext() (pynlpl.formats.folia.ObservationLayer	rightcontext() (pynlpl.formats.folia.TextMarkupGap
method), 700	method), 814
rightcontext() (pynlpl.formats.folia.Original method),	rightcontext() (pynlpl.formats.folia.TextMarkupString
992	method), 824
rightcontext() (pynlpl.formats.folia.Paragraph method),	rightcontext() (pynlpl.formats.folia.TextMarkupStyle
274	method), 835
rightcontext() (pynlpl.formats.folia.Part method), 287	rightcontext() (pynlpl.formats.folia.TimeSegment
rightcontext() (pynlpl.formats.folia.PhonContent	method), 640
method), 511	rightcontext() (pynlpl.formats.folia.TimingLayer
rightcontext() (pynlpl.formats.folia.PosAnnotation	method), 758
method), 445	rightcontext() (pynlpl.formats.folia.Whitespace method),
rightcontext() (pynlpl.formats.folia.Predicate method),	406
582	rightcontext() (pynlpl.formats.folia.Word method), 421
rightcontext() (pynlpl.formats.folia.Quote method), 300	rmse() (in module pynlpl.evaluation), 11
rightcontext() (pynlpl.formats.folia.Reference method),	rmse() (pynlpl.evaluation.OrdinalEvaluation method), 10
313	root() (pynlpl.datatypes.Trie method), 7
rightcontext() (pynlpl.formats.folia.Row method), 326	Row (class in pynlpl.formats.folia), 316
	run() (pynlpl.evaluation.AbstractExperiment method), 9
method), 629	run() (pynlpl.evaluation.ExperimentPool method), 10
rightcontext() (pynlpl.formats.folia.SemanticRolesLayer	S
method), 747	
rightcontext() (pynlpl.formats.folia.SenseAnnotation	sample() (pynlpl.evaluation.AbstractExperiment
method), 478	method), 9
rightcontext() (pynlpl.formats.folia.Sentence method),	save() (pynlpl.formats.folia.Document method), 20
341	save() (pynlpl.formats.sonar.CorpusDocumentX method),
rightcontext() (pynlpl.formats.folia.Sentiment method),	1054

save() (pynlpl.lm.lm.SimpleLanguageModel method), 770 1057 select() (pynlpl.formats.folia.Correction method), 949 select() (pynlpl.formats.folia.Current method), 959 save() (pynlpl.statistics.FrequencyList method), 1063 score() (pynlpl.datatypes.PriorityQueue method), 6 select() (pynlpl.formats.folia.Definition method), 120 score() (pynlpl.evaluation.AbstractExperiment method), select() (pynlpl.formats.folia.DependenciesLayer method), 676 score() (pynlpl.lm.lm.ARPALanguageModel method), select() (pynlpl.formats.folia.Dependency method), 547 (pynlpl.formats.folia.DependencyDependent 1057 select() score() (pynlpl.search.AbstractSearchState method), 1060 method), 782 scoresentence() (pynlpl.lm.client.LMClient method), select() (pynlpl.formats.folia.Description method), 1037 1058 select() (pynlpl.formats.folia.Division method), 133 (pynlpl.lm.lm.SimpleLanguageModel select() (pynlpl.formats.folia.Document method), 20 scoresentence() method), 1057 (pynlpl.formats.folia.DomainAnnotation select() scoresentence() (pynlpl.lm.srilm.SRILM method), 1057 method), 434 (pynlpl.formats.folia.EnddatetimeFeature scoreword() (pynlpl.lm.lm.ARPALanguageModel select() method), 1057 method), 912 searchall() (pynlpl.search.AbstractSearch method), 1059 select() (pynlpl.formats.folia.EntitiesLayer method), 688 (pynlpl.evaluation.WPSParamSearch select() (pynlpl.formats.folia.Entity method), 559 searchbest() method), 10 select() (pynlpl.formats.folia.Entry method), 146 select() (pynlpl.formats.folia.ErrorDetection method), searchbest() (pynlpl.search.AbstractSearch method). 1059 970 searchfirst() (pynlpl.search.AbstractSearch method), 1059 select() (pynlpl.formats.folia.Event method), 159 searchlast() (pynlpl.search.AbstractSearch method), 1059 select() (pynlpl.formats.folia.Example method), 172 searchtop() (pynlpl.search.AbstractSearch method), 1059 select() (pynlpl.formats.folia.Feature method), 868 select() (pynlpl.formats.folia.Figure method), 185 select() (pynlpl.formats.folia.AbstractAnnotationLayer method), 81 select() (pynlpl.formats.folia.Gap method), 196 select() (pynlpl.formats.folia.AbstractElement method), select() (pynlpl.formats.folia.Head method), 209 select() (pynlpl.formats.folia.Headspan method), 794 select() (pynlpl.formats.folia.LangAnnotation method), (pynlpl.formats.folia.AbstractSpanAnnotation select() method), 58 select() (pynlpl.formats.folia.AbstractStructureElement select() (pynlpl.formats.folia.LemmaAnnotation method), method), 44 (pynlpl.formats.folia.AbstractTextMarkup select() (pynlpl.formats.folia.Linebreak method), 222 select() method), 92 select() (pynlpl.formats.folia.List method), 235 (pynlpl.formats.folia.AbstractTokenAnnotation select() (pynlpl.formats.folia.ListItem method), 248 select() method), 69 select() (pynlpl.formats.folia.Metric method), 1048 select() (pynlpl.formats.folia.ActorFeature method), 890 select() (pynlpl.formats.folia.New method), 981 select() (pynlpl.formats.folia.Alignment method), 1014 select() (pynlpl.formats.folia.Note method), 261 select() (pynlpl.formats.folia.AlignReference method), select() (pynlpl.formats.folia.Observation method), 570 1026 select() (pynlpl.formats.folia.ObservationLayer method), select() (pynlpl.formats.folia.Alternative method), 926 700 select() (pynlpl.formats.folia.AlternativeLayers method), select() (pynlpl.formats.folia.Original method), 992 select() (pynlpl.formats.folia.Paragraph method), 274 select() (pynlpl.formats.folia.BegindatetimeFeature select() (pynlpl.formats.folia.Part method), 287 method), 901 select() (pynlpl.formats.folia.PhonContent method), 511 select() (pynlpl.formats.folia.Cell method), 107 select() (pynlpl.formats.folia.PosAnnotation method), select() (pynlpl.formats.folia.Chunk method), 524 445 select() (pynlpl.formats.folia.ChunkingLayer method), select() (pynlpl.formats.folia.Predicate method), 582 select() (pynlpl.formats.folia.Quote method), 300 select() (pynlpl.formats.folia.CoreferenceChain method), select() (pynlpl.formats.folia.Reference method), 313 select() (pynlpl.formats.folia.Row method), 326 select() (pynlpl.formats.folia.SemanticRole method), 629 select() (pynlpl.formats.folia.CoreferenceLayer method), (pynlpl.formats.folia.SemanticRolesLayer select() select() (pynlpl.formats.folia.CoreferenceLink method), method), 747

```
sentences() (pynlpl.formats.folia.Note method), 262
select() (pynlpl.formats.folia.SenseAnnotation method),
         478
                                                          sentences() (pynlpl.formats.folia.Paragraph method), 275
select() (pynlpl.formats.folia.Sentence method), 341
                                                          sentences() (pynlpl.formats.folia.Part method), 288
select() (pynlpl.formats.folia.Sentiment method), 594
                                                          sentences() (pynlpl.formats.folia.Quote method), 301
select() (pynlpl.formats.folia.SentimentLayer method),
                                                          sentences() (pynlpl.formats.folia.Reference method), 314
                                                          sentences() (pynlpl.formats.folia.Row method), 327
select() (pynlpl.formats.folia.Statement method), 605
                                                          sentences() (pynlpl.formats.folia.Sentence method), 341
select() (pynlpl.formats.folia.StatementLayer method),
                                                          sentences() (pynlpl.formats.folia.Table method), 355
                                                          sentences() (pynlpl.formats.folia.TableHead method), 381
select()
            (pynlpl.formats.folia.SubjectivityAnnotation
                                                          sentences() (pynlpl.formats.folia.Term method), 368
         method), 489
                                                          sentences() (pynlpl.formats.folia.Text method), 394
select() (pynlpl.formats.folia.Suggestion method), 1003
                                                          sentences() (pynlpl.formats.folia.Whitespace method),
select() (pynlpl.formats.folia.SynsetFeature method), 879
select() (pynlpl.formats.folia.SyntacticUnit method), 617
                                                          sentences() (pynlpl.formats.folia.Word method), 422
                                                          sentences()
                                                                            (pynlpl.formats.sonar.CorpusDocument
select() (pynlpl.formats.folia.SyntaxLayer method), 735
select() (pynlpl.formats.folia.Table method), 354
                                                                    method), 1054
select() (pynlpl.formats.folia.TableHead method), 380
                                                          sentences()
                                                                          (pynlpl.formats.sonar.CorpusDocumentX
select() (pynlpl.formats.folia.Term method), 367
                                                                    method), 1054
select() (pynlpl.formats.folia.Text method), 393
                                                          Sentiment (class in pynlpl.formats.folia), 585
select() (pynlpl.formats.folia.TextContent method), 501
                                                          SentimentLayer (class in pynlpl.formats.folia), 703
select()
            (pynlpl.formats.folia.TextMarkupCorrection
                                                          sequence() (pynlpl.datatypes.Trie method), 7
         method), 846
                                                          setdoc() (pynlpl.formats.folia.AbstractAnnotationLayer
select() (pynlpl.formats.folia.TextMarkupError method),
                                                                    method), 81
                                                          setdoc() (pynlpl.formats.folia.AbstractElement method),
select() (pynlpl.formats.folia.TextMarkupGap method),
                                                          setdoc()
                                                                     (pynlpl.formats.folia.AbstractSpanAnnotation
select() (pynlpl.formats.folia.TextMarkupString method),
                                                                    method), 58
                                                                    (pynlpl.formats.folia.AbstractStructureElement
                                                          setdoc()
select() (pynlpl.formats.folia.TextMarkupStyle method),
                                                                    method), 45
                                                                          (pynlpl.formats.folia.AbstractTextMarkup
         835
                                                          setdoc()
select() (pynlpl.formats.folia.TimeSegment method), 640
                                                                    method), 92
                                                                    (pynlpl.formats.folia.AbstractTokenAnnotation
select() (pynlpl.formats.folia.TimingLayer method), 759
                                                          setdoc()
select() (pynlpl.formats.folia.Whitespace method), 406
                                                                    method), 69
select() (pynlpl.formats.folia.Word method), 421
                                                          setdoc() (pynlpl.formats.folia.ActorFeature method), 891
SemanticRole (class in pynlpl.formats.folia), 620
                                                          setdoc() (pynlpl.formats.folia.Alignment method), 1015
                                                          setdoc() (pynlpl.formats.folia.AlignReference method),
SemanticRolesLayer (class in pynlpl.formats.folia), 738
sense() (pynlpl.formats.folia.Word method), 422
                                                                    1026
SenseAnnotation (class in pynlpl.formats.folia), 470
                                                          setdoc() (pynlpl.formats.folia.Alternative method), 926
Sentence (class in pynlpl.formats.folia), 329
                                                          setdoc() (pynlpl.formats.folia.AlternativeLayers method),
sentence() (pynlpl.formats.folia.Word method), 422
sentences() (pynlpl.formats.folia.AbstractStructureElement_setdoc()
                                                                        (pynlpl.formats.folia.BegindatetimeFeature
         method), 45
                                                                    method), 902
sentences() (pynlpl.formats.folia.Cell method), 108
                                                          setdoc() (pynlpl.formats.folia.Cell method), 108
sentences() (pynlpl.formats.folia.Definition method), 121
                                                          setdoc() (pynlpl.formats.folia.Chunk method), 524
sentences() (pynlpl.formats.folia.Division method), 134
                                                          setdoc() (pynlpl.formats.folia.ChunkingLayer method),
sentences() (pynlpl.formats.folia.Document method), 20
                                                                    653
                                                          setdoc() (pynlpl.formats.folia.CoreferenceChain method),
sentences() (pynlpl.formats.folia.Entry method), 147
sentences() (pynlpl.formats.folia.Event method), 160
sentences() (pynlpl.formats.folia.Example method), 173
                                                          setdoc() (pynlpl.formats.folia.CoreferenceLayer method),
sentences() (pynlpl.formats.folia.Figure method), 186
                                                          setdoc() (pynlpl.formats.folia.CoreferenceLink method),
sentences() (pynlpl.formats.folia.Head method), 210
sentences() (pynlpl.formats.folia.Linebreak method), 223
                                                          setdoc() (pynlpl.formats.folia.Correction method), 949
sentences() (pynlpl.formats.folia.List method), 236
sentences() (pynlpl.formats.folia.ListItem method), 249
                                                          setdoc() (pynlpl.formats.folia.Current method), 960
```

setdoc() (pynlpl.formats.folia.Definition method), 121 setdoc() (pynlpl.formats.folia.Sentiment method), 594 setdoc() (pynlpl.formats.folia.DependenciesLayer setdoc() (pynlpl.formats.folia.SentimentLayer method), method), 677 setdoc() (pynlpl.formats.folia.Dependency method), 548 setdoc() (pynlpl.formats.folia.Statement method), 606 (pynlpl.formats.folia.DependencyDependent setdoc() (pynlpl.formats.folia.StatementLayer method), setdoc() method), 783 setdoc() (pynlpl.formats.folia.Description method), 1038 setdoc() (pynlpl.formats.folia.SubjectivityAnnotation setdoc() (pynlpl.formats.folia.Division method), 134 method), 490 setdoc() (pynlpl.formats.folia.Suggestion method), 1004 setdoc() (pynlpl.formats.folia.DomainAnnotation method), 435 setdoc() (pynlpl.formats.folia.SynsetFeature method), setdoc() (pynlpl.formats.folia.EnddatetimeFeature method), 913 setdoc() (pynlpl.formats.folia.SyntacticUnit method), 618 setdoc() (pynlpl.formats.folia.EntitiesLayer method), 689 setdoc() (pynlpl.formats.folia.SyntaxLayer method), 736 setdoc() (pynlpl.formats.folia.Entity method), 559 setdoc() (pynlpl.formats.folia.Table method), 355 setdoc() (pynlpl.formats.folia.Entry method), 147 setdoc() (pynlpl.formats.folia.TableHead method), 381 setdoc() (pynlpl.formats.folia.ErrorDetection method), setdoc() (pynlpl.formats.folia.Term method), 368 setdoc() (pynlpl.formats.folia.Text method), 394 setdoc() (pynlpl.formats.folia.Event method), 160 setdoc() (pynlpl.formats.folia.TextContent method), 501 (pynlpl.formats.folia.TextMarkupCorrection setdoc() (pynlpl.formats.folia.Example method), 173 setdoc() setdoc() (pynlpl.formats.folia.Feature method), 869 method), 846 setdoc() (pynlpl.formats.folia.Figure method), 186 setdoc() (pynlpl.formats.folia.TextMarkupError method), setdoc() (pynlpl.formats.folia.Gap method), 197 setdoc() (pynlpl.formats.folia.Head method), 210 setdoc() (pynlpl.formats.folia.TextMarkupGap method), setdoc() (pynlpl.formats.folia.Headspan method), 794 814 setdoc() (pynlpl.formats.folia.LangAnnotation method), (pynlpl.formats.folia.TextMarkupString setdoc() 457 method), 825 setdoc() (pynlpl.formats.folia.LemmaAnnotation setdoc() (pynlpl.formats.folia.TextMarkupStyle method), method), 468 setdoc() (pynlpl.formats.folia.Linebreak method), 223 setdoc() (pynlpl.formats.folia.TimeSegment method), 641 setdoc() (pynlpl.formats.folia.TimingLayer method), 759 setdoc() (pynlpl.formats.folia.List method), 236 setdoc() (pynlpl.formats.folia.ListItem method), 249 setdoc() (pynlpl.formats.folia.Whitespace method), 407 setdoc() (pynlpl.formats.folia.Metric method), 1049 setdoc() (pynlpl.formats.folia.Word method), 422 setdoc() (pynlpl.formats.folia.New method), 982 setdocument() (pynlpl.formats.folia.AbstractAnnotationLayer method), 81 setdoc() (pynlpl.formats.folia.Note method), 262 setdoc() (pynlpl.formats.folia.Observation method), 571 setdocument() (pynlpl.formats.folia.AbstractElement setdoc() (pynlpl.formats.folia.ObservationLayer method), method), 32 700 setdocument() (pynlpl.formats.folia.AbstractSpanAnnotation setdoc() (pynlpl.formats.folia.Original method), 993 method), 58 setdoc() (pynlpl.formats.folia.Paragraph method), 275 setdocument() (pynlpl.formats.folia.AbstractStructureElement setdoc() (pynlpl.formats.folia.Part method), 288 method), 45 setdoc() (pynlpl.formats.folia.PhonContent method), 511 setdocument() (pynlpl.formats.folia.AbstractTextMarkup setdoc() (pynlpl.formats.folia.PosAnnotation method), method), 92 setdocument() (pynlpl.formats.folia.AbstractTokenAnnotation setdoc() (pynlpl.formats.folia.Predicate method), 583 method), 69 setdoc() (pynlpl.formats.folia.Quote method), 301 setdocument() (pynlpl.formats.folia.ActorFeature setdoc() (pynlpl.formats.folia.Reference method), 314 method), 891 setdoc() (pynlpl.formats.folia.Row method), 327 setdocument() (pynlpl.formats.folia.Alignment method), setdoc() (pynlpl.formats.folia.SemanticRole method), 1015 629 setdocument() (pynlpl.formats.folia.AlignReference (pynlpl.formats.folia.SemanticRolesLayer method), 1026 setdoc() method), 747 setdocument() (pynlpl.formats.folia.Alternative method), setdoc() (pynlpl.formats.folia.SenseAnnotation method), 926 setdocument() (pynlpl.formats.folia.AlternativeLayers setdoc() (pynlpl.formats.folia.Sentence method), 342 method), 937

setdocument() (pynlpl.formats.folia.BegindatetimeFeature 249 method), 902 setdocument() (pynlpl.formats.folia.Metric method). setdocument() (pynlpl.formats.folia.Cell method), 108 1049 setdocument() (pynlpl.formats.folia.Chunk method), 524 setdocument() (pynlpl.formats.folia.New method), 982 (pynlpl.formats.folia.ChunkingLayer setdocument() (pynlpl.formats.folia.Note method), 262 setdocument() (pynlpl.formats.folia.Observation method), 653 setdocument() (pynlpl.formats.folia.CoreferenceChain setdocument() method), 571 method), 536 setdocument() (pynlpl.formats.folia.ObservationLayer setdocument() (pynlpl.formats.folia.CoreferenceLayer method), 700 method), 665 setdocument() (pynlpl.formats.folia.Original method), setdocument() (pynlpl.formats.folia.CoreferenceLink 993 setdocument() (pynlpl.formats.folia.Paragraph method), method), 771 setdocument() (pynlpl.formats.folia.Correction method), 949 setdocument() (pynlpl.formats.folia.Part method), 288 setdocument() (pynlpl.formats.folia.Current method), 960 setdocument() (pynlpl.formats.folia.PhonContent setdocument() (pynlpl.formats.folia.Definition method), method), 511 setdocument() (pynlpl.formats.folia.PosAnnotation 121 setdocument() (pynlpl.formats.folia.DependenciesLayer method), 677 setdocument() (pynlpl.formats.folia.Predicate method), (pynlpl.formats.folia.Dependency setdocument() 583 method), 548 setdocument() (pynlpl.formats.folia.Quote method), 301 setdocument() (pynlpl.formats.folia.DependencyDependent setdocument() (pynlpl.formats.folia.Reference method), 314 method), 783 setdocument() (pynlpl.formats.folia.Description method), setdocument() (pynlpl.formats.folia.Row method), 327 (pynlpl.formats.folia.SemanticRole 1038 setdocument() setdocument() (pynlpl.formats.folia.Division method), method), 629 134 setdocument() (pynlpl.formats.folia.SemanticRolesLayer setdocument() (pynlpl.formats.folia.DomainAnnotation method), 747 method), 435 (pynlpl.formats.folia.SenseAnnotation setdocument() setdocument() (pynlpl.formats.folia.EnddatetimeFeature method), 479 method), 913 setdocument() (pynlpl.formats.folia.Sentence method), setdocument() (pynlpl.formats.folia.EntitiesLayer 342 method), 689 setdocument() (pynlpl.formats.folia.Sentiment method), setdocument() (pynlpl.formats.folia.Entity method), 559 594 setdocument() (pynlpl.formats.folia.Entry method), 147 (pynlpl.formats.folia.SentimentLayer setdocument() (pynlpl.formats.folia.ErrorDetection setdocument() method), 712 method), 971 setdocument() (pynlpl.formats.folia.Statement method), setdocument() (pynlpl.formats.folia.Event method), 160 606 setdocument() (pynlpl.formats.folia.Example method), setdocument() (pynlpl.formats.folia.StatementLayer method), 724 173 setdocument() (pynlpl.formats.folia.Feature method), 869 setdocument() (pynlpl.formats.folia.SubjectivityAnnotation setdocument() (pynlpl.formats.folia.Figure method), 186 method), 490 setdocument() (pynlpl.formats.folia.Gap method), 197 setdocument() (pynlpl.formats.folia.Suggestion method), setdocument() (pynlpl.formats.folia.Head method), 210 1004 setdocument() (pynlpl.formats.folia.Headspan method), setdocument() (pynlpl.formats.folia.SynsetFeature 794 method), 880 (pynlpl.formats.folia.LangAnnotation (pynlpl.formats.folia.SyntacticUnit setdocument() setdocument() method), 457 method), 618 (pynlpl.formats.folia.LemmaAnnotation (pynlpl.formats.folia.SyntaxLayer setdocument() setdocument() method), 468 method), 736 setdocument() (pynlpl.formats.folia.Linebreak method), setdocument() (pynlpl.formats.folia.Table method), 355 setdocument() (pynlpl.formats.folia.TableHead method), setdocument() (pynlpl.formats.folia.List method), 236 setdocument() (pynlpl.formats.folia.ListItem method), setdocument() (pynlpl.formats.folia.Term method), 368

- setdocument() (pynlpl.formats.folia.Text method), 394 (pynlpl.formats.folia.TextContent SETONLY setdocument() method), 846 setdocument() (pynlpl.formats.folia.TextMarkupError method), 857 setdocument() (pynlpl.formats.folia.TextMarkupGap method), 814 setdocument() (pynlpl.formats.folia.TextMarkupString method), 825 (pynlpl.formats.folia.TextMarkupStyle setdocument() method), 836 setdocument() (pynlpl.formats.folia.TimeSegment method), 641 setdocument() (pynlpl.formats.folia.TimingLayer method), 759 setdocument() (pynlpl.formats.folia.Whitespace method), setdocument() (pynlpl.formats.folia.Word method), 422 (pynlpl.statistics.HiddenMarkovModel setemission() method), 1063 setimdi() (pynlpl.formats.folia.Document method), 20 SETONLY (pynlpl.formats.folia.AbstractAnnotationLayer SETONLY (pynlpl.formats.folia.Entry attribute), 139 attribute), 75 SETONLY (pynlpl.formats.folia.AbstractElement attribute), 26 SETONLY (pynlpl.formats.folia.AbstractSpanAnnotation attribute), 52 SETONLY (pynlpl.formats.folia.AbstractStructureElement SETONLY (pynlpl.formats.folia.Figure attribute), 178 attribute), 37 SETONLY (pynlpl.formats.folia.AbstractTextMarkup attribute), 86 SETONLY (pynlpl.formats.folia.AbstractTokenAnnotation SETONLY (pynlpl.formats.folia.LangAnnotation attribute), 63 SETONLY (pynlpl.formats.folia.ActorFeature attribute), 885 SETONLY (pynlpl.formats.folia.Alignment attribute), **SETONLY** (pynlpl.formats.folia.AlignReference tribute), 1020 SETONLY (pynlpl.formats.folia.Alternative attribute), SETONLY (pynlpl.formats.folia.AlternativeLayers attribute), 931 **SETONLY** (pynlpl.formats.folia.BegindatetimeFeature attribute), 896 SETONLY (pynlpl.formats.folia.Cell attribute), 100 SETONLY (pynlpl.formats.folia.Chunk attribute), 518 SETONLY (pynlpl.formats.folia.ChunkingLayer tribute), 647 SETONLY (pynlpl.formats.folia.CoreferenceChain attribute), 530 SETONLY (pynlpl.formats.folia.CoreferenceLayer at-
- tribute), 658 (pynlpl.formats.folia.CoreferenceLink attribute), 764 setdocument() (pynlpl.formats.folia.TextMarkupCorrection SETONLY (pynlpl.formats.folia.Correction attribute), SETONLY (pynlpl.formats.folia.Current attribute), 953 SETONLY (pynlpl.formats.folia.Definition attribute), 113 SETONLY (pynlpl.formats.folia.DependenciesLayer attribute), 670 SETONLY (pynlpl.formats.folia.Dependency attribute), SETONLY (pynlpl.formats.folia.DependencyDependent attribute), 776 SETONLY (pynlpl.formats.folia.Description attribute), SETONLY (pynlpl.formats.folia.Division attribute), 126 SETONLY (pynlpl.formats.folia.DomainAnnotation attribute), 429 SETONLY (pynlpl.formats.folia.EnddatetimeFeature attribute), 907 SETONLY (pynlpl.formats.folia.EntitiesLayer attribute), SETONLY (pynlpl.formats.folia.Entity attribute), 553 **SETONLY** (pynlpl.formats.folia.ErrorDetection attribute), 965 SETONLY (pynlpl.formats.folia.Event attribute), 152 SETONLY (pynlpl.formats.folia.Example attribute), 165 SETONLY (pynlpl.formats.folia.Feature attribute), 862 SETONLY (pynlpl.formats.folia.Gap attribute), 191 SETONLY (pynlpl.formats.folia.Head attribute), 202 SETONLY (pynlpl.formats.folia.Headspan attribute), 788 tribute), 451 SETONLY (pynlpl.formats.folia.LemmaAnnotation attribute), 462 SETONLY (pynlpl.formats.folia.Linebreak attribute), 215 SETONLY (pynlpl.formats.folia.List attribute), 228 SETONLY (pynlpl.formats.folia.ListItem attribute), 241 SETONLY (pynlpl.formats.folia.Metric attribute), 1043 SETONLY (pynlpl.formats.folia.New attribute), 976 SETONLY (pynlpl.formats.folia.Note attribute), 254 SETONLY (pynlpl.formats.folia.Observation attribute), 564 SETONLY (pynlpl.formats.folia.ObservationLayer attribute), 694 SETONLY (pynlpl.formats.folia.Original attribute), 987 SETONLY (pynlpl.formats.folia.Paragraph attribute), 267 SETONLY (pynlpl.formats.folia.Part attribute), 280 SETONLY (pynlpl.formats.folia.PhonContent attribute), 506

(pynlpl.formats.folia.PosAnnotation

at-

Index 1161

**SETONLY** 

tribute), 440

- SETONLY (pynlpl.formats.folia.Predicate attribute), 576 SETONLY (pynlpl.formats.folia.Quote attribute), 293 SETONLY (pynlpl.formats.folia.Reference attribute), 306 SETONLY (pynlpl.formats.folia.Row attribute), 319 SETONLY (pynlpl.formats.folia.SemanticRole attribute), SETONLY (pynlpl.formats.folia.SemanticRolesLayer attribute), 741 **SETONLY** (pynlpl.formats.folia.SenseAnnotation attribute), 473 SETONLY (pynlpl.formats.folia.Sentence attribute), 333 SETONLY (pynlpl.formats.folia.Sentiment attribute), 588 (pynlpl.formats.folia.SentimentLayer SETONLY tribute), 705 SETONLY (pynlpl.formats.folia.Statement attribute), 599 (pynlpl.formats.folia.StatementLayer **SETONLY** tribute), 717 SETONLY (pynlpl.formats.folia.SubjectivityAnnotation attribute), 484 SETONLY (pynlpl.formats.folia.Suggestion attribute), SETONLY (pynlpl.formats.folia.SynsetFeature attribute), SETONLY (pynlpl.formats.folia.SyntacticUnit attribute), SETONLY (pynlpl.formats.folia.SyntaxLayer attribute), SETONLY (pynlpl.formats.folia.Table attribute), 347 SETONLY (pynlpl.formats.folia.TableHead attribute), 373 SETONLY (pynlpl.formats.folia.Term attribute), 360 SETONLY (pynlpl.formats.folia.Text attribute), 386 SETONLY (pynlpl.formats.folia.TextContent attribute), SETONLY (pynlpl.formats.folia.TextMarkupCorrection attribute), 840 **SETONLY** (pynlpl.formats.folia.TextMarkupError attribute), 851 (pynlpl.formats.folia.TextMarkupGap **SETONLY** attribute), 808 SETONLY (pynlpl.formats.folia.TextMarkupString attribute), 819 **SETONLY** (pynlpl.formats.folia.TextMarkupStyle attribute), 830 SETONLY (pynlpl.formats.folia.TimeSegment attribute), SETONLY (pynlpl.formats.folia.TimingLayer attribute), SETONLY (pynlpl.formats.folia.Whitespace attribute), SETONLY (pynlpl.formats.folia.Word attribute), 413 setparents() (pynlpl.formats.folia.AbstractAnnotationLayer setparents() (pynlpl.formats.folia.Entry method), 147 method), 81
- method), 32 setparents() (pynlpl.formats.folia.AbstractSpanAnnotation method), 58 setparents() (pynlpl.formats.folia.AbstractStructureElement method), 45 setparents() (pynlpl.formats.folia.AbstractTextMarkup method), 92 setparents() (pynlpl.formats.folia.AbstractTokenAnnotation method), 70 setparents() (pynlpl.formats.folia.ActorFeature method), setparents() (pynlpl.formats.folia.Alignment method), setparents() (pynlpl.formats.folia.AlignReference method), 1026 setparents() (pynlpl.formats.folia.Alternative method), 926 (pynlpl.formats.folia.AlternativeLayers setparents() method), 937 setparents() (pynlpl.formats.folia.BegindatetimeFeature method), 902 setparents() (pynlpl.formats.folia.Cell method), 108 setparents() (pynlpl.formats.folia.Chunk method), 524 (pynlpl.formats.folia.ChunkingLaver setparents() method), 653 setparents() (pynlpl.formats.folia.CoreferenceChain method), 536 (pynlpl.formats.folia.CoreferenceLayer setparents() method), 665 (pynlpl.formats.folia.CoreferenceLink setparents() method), 771 setparents() (pynlpl.formats.folia.Correction method), 949 setparents() (pynlpl.formats.folia.Current method), 960 setparents() (pynlpl.formats.folia.Definition method), 121 (pynlpl.formats.folia.DependenciesLayer setparents() method), 677 setparents() (pynlpl.formats.folia.Dependency method), 548 setparents() (pynlpl.formats.folia.DependencyDependent method), 783 setparents() (pynlpl.formats.folia.Description method), setparents() (pynlpl.formats.folia.Division method), 134 (pynlpl.formats.folia.DomainAnnotation setparents() method), 435 (pynlpl.formats.folia.EnddatetimeFeature setparents() method), 913 setparents() (pynlpl.formats.folia.EntitiesLayer method), 689 setparents() (pynlpl.formats.folia.Entity method), 559

(pynlpl.formats.folia.ErrorDetection

Index 1162

(pynlpl.formats.folia.AbstractElement

setparents()

setparents()

method), 971

setparents() (pynlpl.formats.folia.Event method), 160	736
setparents() (pynlpl.formats.folia.Example method), 173	setparents() (pynlpl.formats.folia.Table method), 355
setparents() (pynlpl.formats.folia.Feature method), 869	setparents() (pynlpl.formats.folia.TableHead method),
setparents() (pynlpl.formats.folia.Figure method), 186	381
setparents() (pynlpl.formats.folia.Gap method), 197	setparents() (pynlpl.formats.folia.Term method), 368
setparents() (pynlpl.formats.folia.Head method), 210	setparents() (pynlpl.formats.folia.Text method), 394
setparents() (pynlpl.formats.folia.Headspan method), 794	setparents() (pynlpl.formats.folia.TextContent method).
setparents() (pynlpl.formats.folia.LangAnnotation	501
method), 457	setparents() (pynlpl.formats.folia.TextMarkupCorrection
setparents() (pynlpl.formats.folia.LemmaAnnotation	method), 846
method), 468	setparents() (pynlpl.formats.folia.TextMarkupError
setparents() (pynlpl.formats.folia.Linebreak method), 223	method), 857
setparents() (pynlpl.formats.folia.List method), 226	setparents() (pynlpl.formats.folia.TextMarkupGap
setparents() (pynlpl.formats.folia.ListItem method), 249	method), 814
setparents() (pynlpl.formats.folia.Metric method), 1049	setparents() (pynlpl.formats.folia.TextMarkupString
setparents() (pynlpl.formats.folia.New method), 982	method), 825
setparents() (pynlpl.formats.folia.Note method), 262	setparents() (pynlpl.formats.folia.TextMarkupStyle
setparents() (pynlpl.formats.folia.Observation method),	method), 836
571	setparents() (pynlpl.formats.folia.TimeSegment method),
setparents() (pynlpl.formats.folia.ObservationLayer	641
method), 700	setparents() (pynlpl.formats.folia.TimingLayer method),
setparents() (pynlpl.formats.folia.Original method), 993	759
setparents() (pynlpl.formats.folia.Paragraph method), 275	setparents() (pynlpl.formats.folia.Whitespace method),
setparents() (pynlpl.formats.folia.Part method), 288	407
setparents() (pynlpl.formats.folia.PhonContent method),	setparents() (pynlpl.formats.folia.Word method), 422
511	setphon() (pynlpl.formats.folia.PhonContent method),
setparents() (pynlpl.formats.folia.PosAnnotation	511
method), 446	setspan() (pynlpl.formats.folia.AbstractSpanAnnotation
setparents() (pynlpl.formats.folia.Predicate method), 583	method), 59
setparents() (pynlpl.formats.folia.Quote method), 301	setspan() (pynlpl.formats.folia.Chunk method), 525
setparents() (pynlpl.formats.folia.Reference method), 314	setspan() (pynlpl.formats.folia.CoreferenceChain
setparents() (pynlpl.formats.folia.Row method), 327	method), 536
setparents() (pynlpl.formats.folia.SemanticRole method),	setspan() (pynlpl.formats.folia.CoreferenceLink method),
629	771
setparents() (pynlpl.formats.folia.SemanticRolesLayer	setspan() (pynlpl.formats.folia.Dependency method), 548
method), 747	setspan() (pynlpl.formats.folia.DependencyDependent
setparents() (pynlpl.formats.folia.SenseAnnotation	method), 783
method), 479	setspan() (pynlpl.formats.folia.Entity method), 559
setparents() (pynlpl.formats.folia.Sentence method), 342	setspan() (pynlpl.formats.folia.Headspan method), 794
setparents() (pynlpl.formats.folia.Sentiment method), 542	setspan() (pynlpl.formats.folia.Observation method), 571
setparents() (pynlpl.formats.folia.SentimentLayer	setspan() (pynlpl.formats.folia.Predicate method), 583
method), 712	setspan() (pynlpl.formats.folia.SemanticRole method),
setparents() (pynlpl.formats.folia.Statement method), 606	629
* * * *	setspan() (pynlpl.formats.folia.Sentiment method), 594
- · · · · · · · · · · · · · · · · · · ·	
method), 724	setspan() (pynlpl.formats.folia.Statement method), 606
setparents() (pynlpl.formats.folia.SubjectivityAnnotation	setspan() (pynlpl.formats.folia.SyntacticUnit method),
method), 490	618
setparents() (pynlpl.formats.folia.Suggestion method),	setspan() (pynlpl.formats.folia.TimeSegment method),
1004	641
setparents() (pynlpl.formats.folia.SynsetFeature method),	settext() (pynlpl.formats.folia.AbstractAnnotationLayer
880	method), 82
setparents() (pynlpl.formats.folia.SyntacticUnit method),	settext() (pynlpl.formats.folia.AbstractElement method),
618	32
setparents() (pynlpl.formats.folia.SyntaxLayer method),	settext() (pynlpl.formats.folia.AbstractSpanAnnotation

	method), 59	settext() (pynlpl.formats.folia.LemmaAnnotat	ion
settext()	(pynlpl.formats.folia.AbstractStructureElement	method), 468	
	method), 45	settext() (pynlpl.formats.folia.Linebreak method), 223	
settext()	(pynlpl.formats.folia.AbstractTextMarkup	settext() (pynlpl.formats.folia.List method), 236	
	method), 92	settext() (pynlpl.formats.folia.ListItem method), 249	
settext()	(pynlpl.formats.folia.AbstractTokenAnnotation	settext() (pynlpl.formats.folia.Metric method), 1049	
	method), 70	settext() (pynlpl.formats.folia.New method), 982	
settext()	(pynlpl.formats.folia.ActorFeature method), 891	settext() (pynlpl.formats.folia.Note method), 262	
settext()	(pynlpl.formats.folia.Alignment method), 1015	settext() (pynlpl.formats.folia.Observation method), 5'	71
settext()	(pynlpl.formats.folia.AlignReference method), 1026	settext() (pynlpl.formats.folia.ObservationLayer metho 701	od),
settext()	(pynlpl.formats.folia.Alternative method), 926	settext() (pynlpl.formats.folia.Original method), 993	
settext()	(pynlpl.formats.folia.AlternativeLayers method),	settext() (pynlpl.formats.folia.Paragraph method), 275	
	938	settext() (pynlpl.formats.folia.Part method), 288	
settext()	(pynlpl.formats.folia.BegindatetimeFeature	settext() (pynlpl.formats.folia.PhonContent method), 5	
	method), 902	settext() (pynlpl.formats.folia.PosAnnotation method	od),
	(pynlpl.formats.folia.Cell method), 108	446	
	(pynlpl.formats.folia.Chunk method), 525	settext() (pynlpl.formats.folia.Predicate method), 583	
settext()	(pynlpl.formats.folia.ChunkingLayer method),	settext() (pynlpl.formats.folia.Quote method), 301	
	653	settext() (pynlpl.formats.folia.Reference method), 314	
settext()	(pynlpl.formats.folia.CoreferenceChain	settext() (pynlpl.formats.folia.Row method), 327	•
	method), 536	settext() (pynlpl.formats.folia.SemanticRole method	od),
settext()	(pynlpl.formats.folia.CoreferenceLayer method),	629	
	665	settext() (pynlpl.formats.folia.SemanticRolesLa	yer
settext()	(pynlpl.formats.folia.CoreferenceLink method),	method), 748	. 1\
cottovt()	771 (nynln) formats folio Correction method), 040	settext() (pynlpl.formats.folia.SenseAnnotation method 479	)a),
	(pynlpl.formats.folia.Correction method), 949		
	(pynlpl.formats.folia.Current method), 960 (pynlpl.formats.folia.Definition method), 121	settext() (pynlpl.formats.folia.Sentence method), 342 settext() (pynlpl.formats.folia.Sentiment method), 594	
	(pynlpl.formats.folia.DependenciesLayer	settext() (pynlpl.formats.folia.Sentiment Inethod), 394 settext() (pynlpl.formats.folia.SentimentLayer method	
settext()	method), 677	712	σ,
cettevt()	(pynlpl.formats.folia.Dependency method), 548	settext() (pynlpl.formats.folia.Statement method), 606	
settext()	(pynlpl.formats.folia.DependencyDependent	settext() (pynlpl.formats.folia.StatementLayer method); settext() (pynlpl.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.formats.form	
	method), 783	724	
settext()	(pynlpl.formats.folia.Description method), 1038 (pynlpl.formats.folia.Division method), 134	settext() (pynlpl.formats.folia.SubjectivityAnnotat method), 490	
settext()	(pynlpl.formats.folia.DomainAnnotation method), 435	settext() (pynlpl.formats.folia.Suggestion method), 10	
cottoxt()	(pynlpl.formats.folia.EnddatetimeFeature	settext() (pynlpl.formats.folia.SynsetFeature methors 880	σ,
settext()	method), 913	settext() (pynlpl.formats.folia.SyntacticUnit method),	618
cettevt()	(pynlpl.formats.folia.EntitiesLayer method), 689	settext() (pynlpl.formats.folia.SyntaxLayer method), 7	
	(pynlpl.formats.folia.Entity method), 559	settext() (pynlpl.formats.folia.Table method), 355	50
	(pynlpl.formats.folia.Entry method), 147	settext() (pynlpl.formats.folia.TableHead method), 38	1
settext()		settext() (pynlpl.formats.folia.Term method), 368	L
Sette Att()	971	settext() (pynlpl.formats.folia.Text method), 394	
settext()	(pynlpl.formats.folia.Event method), 160	settext() (pynlpl.formats.folia.TextContent method), 5	01
	(pynlpl.formats.folia.Example method), 173	settext() (pynlpl.formats.folia.TextMarkupCorrect	
	(pynlpl.formats.folia.Feature method), 869	method), 847	
	(pynlpl.formats.folia.Figure method), 186	settext() (pynlpl.formats.folia.TextMarkupError metho	od).
	(pynlpl.formats.folia.Gap method), 197	857	,,
	(pynlpl.formats.folia.Head method), 210	settext() (pynlpl.formats.folia.TextMarkupGap metho	od).
	(pynlpl.formats.folia.Headspan method), 794	815	
	(pynlpl.formats.folia.LangAnnotation method),	settext() (pynlpl.formats.folia.TextMarkupStr	ing
	457	method), 825	

- settext() (pynlpl.formats.folia.TextMarkupStyle method),
- (pynlpl.formats.folia.TimeSegment method), settext()
- settext() (pynlpl.formats.folia.TimingLayer method), 759 settext() (pynlpl.formats.folia.Whitespace method), 407
- settext() (pynlpl.formats.folia, Word method), 422
- settransitions() (pynlpl.statistics.MarkovChain method), 1063
- SimpleLanguageModel (class in pynlpl.lm.lm), 1057
- size() (pynlpl.datatypes.Trie method), 7
- size() (pynlpl.statistics.MarkovChain method), 1064
- SPEAKABLE (pynlpl.formats.folia.AbstractAnnotationLay&PEAKABLE attribute), 75
- SPEAKABLE (pynlpl.formats.folia.AbstractElement attribute), 26
- SPEAKABLE (pynlpl.formats.folia.AbstractSpanAnnotatio&PEAKABLE (pynlpl.formats.folia.ErrorDetection atattribute), 52 tribute), 965
- SPEAKABLE (pynlpl.formats.folia.AbstractStructureEleme@PEAKABLE (pynlpl.formats.folia.Event attribute), 152 attribute), 37
- SPEAKABLE (pynlpl.formats.folia.AbstractTextMarkup attribute), 86
- SPEAKABLE (pynlpl.formats.folia.AbstractTokenAnnotation 862 attribute), 63
- SPEAKABLE (pynlpl.formats.folia.ActorFeature tribute), 885
- SPEAKABLE (pynlpl.formats.folia.Alignment attribute),
- SPEAKABLE (pynlpl.formats.folia.AlignReference attribute), 1020
- **SPEAKABLE** (pynlpl.formats.folia.Alternative attribute), 919
- SPEAKABLE (pynlpl.formats.folia.AlternativeLayers attribute), 931
- attribute), 896
- SPEAKABLE (pynlpl.formats.folia.Cell attribute), 100
- SPEAKABLE (pynlpl.formats.folia.Chunk attribute), 518
- SPEAKABLE (pynlpl.formats.folia.ChunkingLayer attribute), 647
- (pynlpl.formats.folia.CoreferenceChain **SPEAKABLE** attribute), 530
- SPEAKABLE (pynlpl.formats.folia.CoreferenceLayer attribute), 658
- SPEAKABLE (pynlpl.formats.folia.CoreferenceLink attribute), 764
- SPEAKABLE (pynlpl.formats.folia.Correction attribute),
- SPEAKABLE (pynlpl.formats.folia.Current attribute),
- SPEAKABLE (pynlpl.formats.folia.Definition attribute),
- SPEAKABLE (pynlpl.formats.folia.DependenciesLayer attribute), 670

- (pynlpl.formats.folia.Dependency SPEAKABLE tribute), 541
- SPEAKABLE (pynlpl.formats.folia.DependencyDependent attribute), 776
- (pynlpl.formats.folia.Description **SPEAKABLE** tribute), 1032
- SPEAKABLE (pynlpl.formats.folia.Division attribute), 126
- SPEAKABLE (pynlpl.formats.folia.DomainAnnotation attribute), 429
- SPEAKABLE (pynlpl.formats.folia.EnddatetimeFeature attribute), 907
- (pynlpl.formats.folia.EntitiesLayer tribute), 682
- SPEAKABLE (pynlpl.formats.folia.Entity attribute), 553
- SPEAKABLE (pynlpl.formats.folia.Entry attribute), 139
- SPEAKABLE (pynlpl.formats.folia.Example attribute),
  - 165 SPEAKABLE (pynlpl.formats.folia.Feature attribute),
  - SPEAKABLE (pynlpl.formats.folia.Figure attribute). 178
  - SPEAKABLE (pynlpl.formats.folia.Gap attribute), 191
  - SPEAKABLE (pynlpl.formats.folia.Head attribute), 202
  - SPEAKABLE (pynlpl.formats.folia.Headspan attribute),
  - SPEAKABLE (pynlpl.formats.folia.LangAnnotation attribute), 451
  - SPEAKABLE (pynlpl.formats.folia.LemmaAnnotation attribute), 462
  - SPEAKABLE (pynlpl.formats.folia.Linebreak attribute), 215
- SPEAKABLE (pynlpl.formats.folia.BegindatetimeFeature SPEAKABLE (pynlpl.formats.folia.List attribute), 228
  - SPEAKABLE (pynlpl.formats.folia.ListItem attribute), 241
  - SPEAKABLE (pynlpl.formats.folia.Metric attribute), 1043
  - SPEAKABLE (pynlpl.formats.folia.New attribute), 976
  - SPEAKABLE (pynlpl.formats.folia.Note attribute), 254
  - SPEAKABLE (pynlpl.formats.folia.Observation tribute), 564
  - SPEAKABLE (pynlpl.formats.folia.ObservationLayer attribute), 694
  - SPEAKABLE (pynlpl.formats.folia.Original attribute),
  - SPEAKABLE (pynlpl.formats.folia.Paragraph attribute),
  - SPEAKABLE (pynlpl.formats.folia.Part attribute), 280
  - (pynlpl.formats.folia.PhonContent SPEAKABLE tribute), 506
  - SPEAKABLE (pynlpl.formats.folia.PosAnnotation attribute), 440

tribute), 399 SPEAKABLE (pynlpl.formats.folia.Predicate attribute), SPEAKABLE (pynlpl.formats.folia.Word attribute), 413 576 SPEAKABLE (pynlpl.formats.folia.Quote attribute), 293 specificity() (pynlpl.evaluation.ClassEvaluation method), SPEAKABLE (pynlpl.formats.folia.Reference attribute), speech speaker() (pynlpl.formats.folia.AbstractAnnotationLayer SPEAKABLE (pynlpl.formats.folia.Row attribute), 319 method), 82 **SPEAKABLE** (pynlpl.formats.folia.SemanticRole speech speaker() (pynlpl.formats.folia.AbstractElement attribute), 623 method), 32 speech\_speaker() (pynlpl.formats.folia.AbstractSpanAnnotation SPEAKABLE (pynlpl.formats.folia.SemanticRolesLayer attribute), 741 method), 59 SPEAKABLE (pynlpl.formats.folia.SenseAnnotation at $speech\_speaker() \, (pynlpl.formats.folia. AbstractStructure Element$ tribute), 473 method), 45 SPEAKABLE (pynlpl.formats.folia.Sentence attribute), speech\_speaker() (pynlpl.formats.folia.AbstractTextMarkup method), 92 SPEAKABLE (pynlpl.formats.folia.Sentiment attribute), speech\_speaker() (pynlpl.formats.folia.AbstractTokenAnnotation method), 70 SPEAKABLE (pynlpl.formats.folia.SentimentLayer atspeech\_speaker() (pynlpl.formats.folia.ActorFeature tribute), 705 method), 891 speech\_speaker() SPEAKABLE (pynlpl.formats.folia.Statement attribute), (pynlpl.formats.folia.Alignment method), 1015 SPEAKABLE (pynlpl.formats.folia.StatementLayer atspeech\_speaker() (pynlpl.formats.folia.AlignReference tribute), 717 method), 1027 SPEAKABLE (pynlpl.formats.folia.SubjectivityAnnotation speech\_speaker() (pynlpl.formats.folia.Alternative attribute), 484 method), 927 **SPEAKABLE** (pynlpl.formats.folia.Suggestion speech speaker() (pynlpl.formats.folia.AlternativeLayers attribute), 998 method), 938 **SPEAKABLE** (pynlpl.formats.folia.SynsetFeature speech\_speaker() (pynlpl.formats.folia.BegindatetimeFeature method), 902 attribute), 874 speech\_speaker() (pynlpl.formats.folia.Cell method), 108 SPEAKABLE (pynlpl.formats.folia.SyntacticUnit speech\_speaker() (pynlpl.formats.folia.Chunk method), tribute), 611 (pynlpl.formats.folia.SyntaxLayer SPEAKABLE at-525 tribute), 729 speech\_speaker() (pynlpl.formats.folia.ChunkingLayer SPEAKABLE (pynlpl.formats.folia.Table attribute), 347 method), 654 SPEAKABLE (pynlpl.formats.folia.TableHead attribute), speech\_speaker() (pynlpl.formats.folia.CoreferenceChain method), 536 SPEAKABLE (pynlpl.formats.folia.Term attribute), 360 speech\_speaker() (pynlpl.formats.folia.CoreferenceLayer SPEAKABLE (pynlpl.formats.folia.Text attribute), 386 method), 665 SPEAKABLE (pynlpl.formats.folia.TextContent speech\_speaker() (pynlpl.formats.folia.CoreferenceLink tribute), 495 method), 771 SPEAKABLE (pynlpl.formats.folia.TextMarkupCorrection speech\_speaker() (pynlpl.formats.folia.Correction attribute), 840 method), 950 speech\_speaker() (pynlpl.formats.folia.Current method), SPEAKABLE (pynlpl.formats.folia.TextMarkupError attribute), 851 SPEAKABLE (pynlpl.formats.folia.TextMarkupGap at-(pynlpl.formats.folia.Definition speech\_speaker() tribute), 808 method), 121 (pynlpl.formats.folia.TextMarkupString speech\_speaker() (pynlpl.formats.folia.DependenciesLayer **SPEAKABLE** attribute), 819 method), 677 SPEAKABLE (pynlpl.formats.folia.TextMarkupStyle atspeech\_speaker() (pynlpl.formats.folia.Dependency method), 548 tribute), 830 **SPEAKABLE** (pynlpl.formats.folia.TimeSegment speech\_speaker() (pynlpl.formats.folia.DependencyDependent attribute), 634 method), 783 SPEAKABLE (pynlpl.formats.folia.TimingLayer (pynlpl.formats.folia.Description atspeech speaker() tribute), 752 method), 1038 **SPEAKABLE** (pynlpl.formats.folia.Whitespace speech speaker() (pynlpl.formats.folia.Division method),

method), 583 134 speech\_speaker() (pynlpl.formats.folia.DomainAnnotation speech\_speaker() (pynlpl.formats.folia.Quote method), method), 435 speech\_speaker() (pynlpl.formats.folia.EnddatetimeFeature speech\_speaker() (pynlpl.formats.folia.Reference method), 913 method), 314 speech speaker() (pynlpl.formats.folia.EntitiesLayer speech\_speaker() (pynlpl.formats.folia.Row method), 327 (pynlpl.formats.folia.SemanticRole method), 689 speech speaker() speech\_speaker() (pynlpl.formats.folia.Entity method), method), 630 560 speech\_speaker() (pynlpl.formats.folia.SemanticRolesLayer speech\_speaker() (pynlpl.formats.folia.Entry method), method), 748 147 speech\_speaker() (pynlpl.formats.folia.SenseAnnotation (pynlpl.formats.folia.ErrorDetection method), 479 speech\_speaker() (pynlpl.formats.folia.Sentence method), 971 speech\_speaker() speech\_speaker() (pynlpl.formats.folia.Event method), method), 342 speech\_speaker() (pynlpl.formats.folia.Sentiment speech\_speaker() (pynlpl.formats.folia.Example method), method), 595 speech\_speaker() (pynlpl.formats.folia.SentimentLayer 173 speech speaker() (pynlpl.formats.folia.Feature method), method), 712 speech\_speaker() (pynlpl.formats.folia.Statement speech speaker() (pynlpl.formats.folia.Figure method), method), 606 (pynlpl.formats.folia.StatementLayer 186 speech\_speaker() speech speaker() (pynlpl.formats.folia.Gap method), 197 method), 724 speech\_speaker() (pynlpl.formats.folia.Head method), speech\_speaker() (pynlpl.formats.folia.SubjectivityAnnotation method), 490 speech\_speaker() speech\_speaker() (pynlpl.formats.folia.Headspan (pynlpl.formats.folia.Suggestion method), 795 method), 1004 speech\_speaker() (pynlpl.formats.folia.LangAnnotation speech\_speaker() (pynlpl.formats.folia.SynsetFeature method), 457 method), 880 speech\_speaker() (pynlpl.formats.folia.LemmaAnnotation (pynlpl.formats.folia.SyntacticUnit speech\_speaker() method), 468 method), 618 speech\_speaker() (pynlpl.formats.folia.Linebreak speech\_speaker() (pynlpl.formats.folia.SyntaxLayer method), 223 method), 736 speech\_speaker() (pynlpl.formats.folia.List method), 236 (pynlpl.formats.folia.Table method), speech\_speaker() speech\_speaker() (pynlpl.formats.folia.ListItem method), 355 (pynlpl.formats.folia.TableHead 249 speech speaker() speech\_speaker() (pynlpl.formats.folia.Metric method), method), 381 1049 speech speaker() (pynlpl.formats.folia.Term method), speech\_speaker() (pynlpl.formats.folia.New method), 982 speech speaker() (pynlpl.formats.folia.Note method), speech\_speaker() (pynlpl.formats.folia.Text method), 394 (pynlpl.formats.folia.TextContent 262 speech\_speaker() speech speaker() (pynlpl.formats.folia.Observation method), 501 method), 571 speech speaker() (pynlpl.formats.folia.TextMarkupCorrection speech speaker() (pynlpl.formats.folia.ObservationLayer method), 847 method), 701 speech\_speaker() (pynlpl.formats.folia.TextMarkupError speech\_speaker() (pynlpl.formats.folia.Original method), method), 857 993 speech\_speaker() (pynlpl.formats.folia.TextMarkupGap (pynlpl.formats.folia.Paragraph speech\_speaker() method), 815 method), 275 speech\_speaker() (pynlpl.formats.folia.TextMarkupString speech\_speaker() (pynlpl.formats.folia.Part method), 288 method), 825 (pynlpl.formats.folia.PhonContent speech\_speaker() (pynlpl.formats.folia.TextMarkupStyle speech\_speaker() method), 836 method), 512 (pynlpl.formats.folia.PosAnnotation (pynlpl.formats.folia.TimeSegment speech\_speaker() speech speaker() method), 446 method), 641 (pynlpl.formats.folia.Predicate (pynlpl.formats.folia.TimingLayer speech speaker() speech speaker()

method), 759	method), 435
speech_speaker() (pynlpl.formats.folia.Whitesp method), 407	pace speech_src() (pynlpl.formats.folia.EnddatetimeFeature method), 913
speech_speaker() (pynlpl.formats.folia.Word meth 422	od), speech_src() (pynlpl.formats.folia.EntitiesLayer method), 689
<pre>speech_src() (pynlpl.formats.folia.AbstractAnnotation</pre>	aLayerspeech_src() (pynlpl.formats.folia.Entity method), 560 speech_src() (pynlpl.formats.folia.Entry method), 147
speech_src() (pynlpl.formats.folia.AbstractElen method), 32	
<i>"</i>	tation speech_src() (pynlpl.formats.folia.Event method), 160 speech_src() (pynlpl.formats.folia.Example method), 173
	lementpeech_src() (pynlpl.formats.folia.Feature method), 869 speech_src() (pynlpl.formats.folia.Figure method), 186
speech_src() (pynlpl.formats.folia.AbstractTextMar method), 93	
	otationspeech_src() (pynlpl.formats.folia.Headspan method), 795
speech_src() (pynlpl.formats.folia.ActorFeature meth 891	
speech_src() (pynlpl.formats.folia.Alignment meth 1016	<i>"</i>
speech_src() (pynlpl.formats.folia.AlignReference method), 1027	
speech_src() (pynlpl.formats.folia.Alternative meth	
speech_src() (pynlpl.formats.folia.AlternativeLa method), 938	
speech_src() (pynlpl.formats.folia.BegindatetimeFea method), 902	
speech_src() (pynlpl.formats.folia.Cell method), 108	571
speech_src() (pynlpl.formats.folia.Chunk method), 52 speech_src() (pynlpl.formats.folia.ChunkingLa	
method), 654	speech_src() (pynlpl.formats.folia.Original method), 993
speech_src() (pynlpl.formats.folia.CoreferenceCl method), 536	nain speech_src() (pynlpl.formats.folia.Paragraph method), 275
speech_src() (pynlpl.formats.folia.CoreferenceLa	_, -, -
method), 666	speech_src() (pynlpl.formats.folia.PhonContent method),
speech_src() (pynlpl.formats.folia.CoreferenceI	
method), 771 speech_src() (pynlpl.formats.folia.Correction meth	speech_src() (pynlpl.formats.folia.PosAnnotation od), method), 446
950	speech_src() (pynlpl.formats.folia.Predicate method), 583
speech_src() (pynlpl.formats.folia.Current method), 9	
speech_src() (pynlpl.formats.folia.Definition meth 121	od), speech_src() (pynlpl.formats.folia.Reference method), 314
speech_src() (pynlpl.formats.folia.DependenciesLamethod), 677	ayer speech_src() (pynlpl.formats.folia.Row method), 327 speech_src() (pynlpl.formats.folia.SemanticRole
speech_src() (pynlpl.formats.folia.Dependency meth 548	od), method), 630 speech_src() (pynlpl.formats.folia.SemanticRolesLayer
speech_src() (pynlpl.formats.folia.DependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyDependencyD	- · · · · · · · · · · · · · · · · · · ·
speech_src() (pynlpl.formats.folia.Description meth 1038	
<pre>speech_src() (pynlpl.formats.folia.Division method), speech_src() (pynlpl.formats.folia.DomainAnnota</pre>	speech_src() (pynlpl.formats.folia.Sentiment method),

- speech\_src() (pynlpl.formats.folia.SentimentLayer method), 713
- speech\_src() (pynlpl.formats.folia.Statement method), 606
- speech\_src() (pynlpl.formats.folia.StatementLayer method), 724
- speech\_src() (pynlpl.formats.folia.SubjectivityAnnotation method), 490
- $\begin{array}{c} speech\_src() \ \ (pynlpl.formats.folia.Suggestion \ \ method), \\ 1004 \end{array}$
- speech\_src() (pynlpl.formats.folia.SynsetFeature method), 880
- speech\_src() (pynlpl.formats.folia.SyntacticUnit method), 618
- speech\_src() (pynlpl.formats.folia.SyntaxLayer method), 736
- speech\_src() (pynlpl.formats.folia.Table method), 355 speech\_src() (pynlpl.formats.folia.TableHead method),
- speech\_src() (pynlpl.formats.folia.Term method), 368 speech\_src() (pynlpl.formats.folia.Text method), 394
- speech\_src() (pynlpl.formats.folia.TextContent method),
  502
- speech\_src() (pynlpl.formats.folia.TextMarkupCorrection method), 847
- speech\_src() (pynlpl.formats.folia.TextMarkupError method), 857
- speech\_src() (pynlpl.formats.folia.TextMarkupGap method), 815
- speech\_src() (pynlpl.formats.folia.TextMarkupString method), 825
- speech\_src() (pynlpl.formats.folia.TextMarkupStyle method), 836
- speech\_src() (pynlpl.formats.folia.TimeSegment method), 641
- speech\_src() (pynlpl.formats.folia.TimingLayer method), 760
- speech\_src() (pynlpl.formats.folia.Whitespace method), 407
- speech\_src() (pynlpl.formats.folia.Word method), 422 split() (pynlpl.formats.folia.Word method), 422
- split\_sentences() (in module pynlpl.textprocessors), 1066 splitword() (pynlpl.formats.folia.Sentence method), 342
- SRILM (class in pynlpl.lm.srilm), 1057
- SRILMException, 1058
- start() (pynlpl.evaluation.AbstractExperiment method), 9 start() (pynlpl.evaluation.ExperimentPool method), 10
- start() (pynlpl.evaluation.ExperimentPool method), 10 startcommand() (pynlpl.evaluation.AbstractExperiment method), 9
- Statement (class in pynlpl.formats.folia), 597
- $Statement Layer\ (class\ in\ pynlpl. formats. folia),\ 714$
- stddev() (in module pynlpl.statistics), 1064
- StochasticBeamSearch (class in pynlpl.search), 1060
- stochasticprune() (pynlpl.datatypes.PriorityQueue

- method), 6
- stricttext() (pynlpl.formats.folia.AbstractAnnotationLayer method), 82
- stricttext() (pynlpl.formats.folia.AbstractElement method), 32
- stricttext() (pynlpl.formats.folia.AbstractSpanAnnotation method), 59
- stricttext() (pynlpl.formats.folia.AbstractStructureElement method), 46
- stricttext() (pynlpl.formats.folia.AbstractTextMarkup method), 93
- stricttext() (pynlpl.formats.folia.AbstractTokenAnnotation method), 70
- stricttext() (pynlpl.formats.folia.ActorFeature method),
- stricttext() (pynlpl.formats.folia.Alignment method), 1016
- stricttext() (pynlpl.formats.folia.AlignReference method),
- stricttext() (pynlpl.formats.folia.Alternative method), 927 stricttext() (pynlpl.formats.folia.AlternativeLayers method), 938
- stricttext() (pynlpl.formats.folia.BegindatetimeFeature method), 903
- stricttext() (pynlpl.formats.folia.Cell method), 109
- stricttext() (pynlpl.formats.folia.Chunk method), 525
- stricttext() (pynlpl.formats.folia.ChunkingLayer method), 654
- stricttext() (pynlpl.formats.folia.CoreferenceChain method), 537
- stricttext() (pynlpl.formats.folia.CoreferenceLayer method), 666
- stricttext() (pynlpl.formats.folia.CoreferenceLink method), 771
- stricttext() (pynlpl.formats.folia.Correction method), 950 stricttext() (pynlpl.formats.folia.Current method), 960
- stricttext() (pynlpl.formats.folia.Definition method), 122
- stricttext() (pynlpl.formats.folia.DependenciesLayer method), 677
- stricttext() (pynlpl.formats.folia.Dependency method), 548
- stricttext() (pynlpl.formats.folia.DependencyDependent method), 783
- stricttext() (pynlpl.formats.folia.Description method), 1038
- stricttext() (pynlpl.formats.folia.Division method), 135
- stricttext() (pynlpl.formats.folia.DomainAnnotation method), 435
- stricttext() (pynlpl.formats.folia.EnddatetimeFeature method), 914
- stricttext() (pynlpl.formats.folia.EntitiesLayer method), 689
- stricttext() (pynlpl.formats.folia.Entity method), 560 stricttext() (pynlpl.formats.folia.Entry method), 148
- Index 1169

```
stricttext() (pynlpl.formats.folia.ErrorDetection method),
                                                                     618
                                                           stricttext() (pynlpl.formats.folia.SyntaxLayer method),
stricttext() (pynlpl.formats.folia.Event method), 161
stricttext() (pynlpl.formats.folia.Example method), 174
                                                           stricttext() (pynlpl.formats.folia.Table method), 355
stricttext() (pynlpl.formats.folia.Feature method), 869
                                                           stricttext() (pynlpl.formats.folia.TableHead method), 381
stricttext() (pynlpl.formats.folia.Figure method), 187
                                                           stricttext() (pynlpl.formats.folia.Term method), 368
stricttext() (pynlpl.formats.folia.Gap method), 198
                                                           stricttext() (pynlpl.formats.folia.Text method), 394
stricttext() (pynlpl.formats.folia.Head method), 211
                                                           stricttext() (pynlpl.formats.folia.TextContent method),
stricttext() (pynlpl.formats.folia.Headspan method), 795
stricttext()
                   (pynlpl.formats.folia.LangAnnotation
                                                           stricttext()
                                                                       (pynlpl.formats.folia.TextMarkupCorrection
          method), 457
                                                                     method), 847
                 (pynlpl.formats.folia.LemmaAnnotation
                                                                             (pynlpl.formats.folia.TextMarkupError
stricttext()
                                                           stricttext()
          method), 468
                                                                     method), 858
stricttext() (pynlpl.formats.folia.Linebreak method), 223
                                                           stricttext()
                                                                              (pynlpl.formats.folia.TextMarkupGap
stricttext() (pynlpl.formats.folia.List method), 237
                                                                     method), 815
stricttext() (pynlpl.formats.folia.ListItem method), 250
                                                           stricttext()
                                                                            (pynlpl.formats.folia.TextMarkupString
stricttext() (pynlpl.formats.folia.Metric method), 1049
                                                                     method), 826
stricttext() (pynlpl.formats.folia.New method), 982
                                                                              (pynlpl.formats.folia.TextMarkupStyle
                                                           stricttext()
stricttext() (pynlpl.formats.folia.Note method), 263
                                                                     method), 836
stricttext() (pynlpl.formats.folia.Observation method),
                                                           stricttext() (pynlpl.formats.folia.TimeSegment method),
         572
                                                                     642
stricttext()
                  (pynlpl.formats.folia.ObservationLayer
                                                           stricttext() (pynlpl.formats.folia.TimingLayer method),
          method), 701
stricttext() (pynlpl.formats.folia.Original method), 993
                                                           stricttext() (pynlpl.formats.folia.Whitespace method),
stricttext() (pynlpl.formats.folia.Paragraph method), 276
stricttext() (pynlpl.formats.folia.Part method), 289
                                                           stricttext() (pynlpl.formats.folia.Word method), 422
stricttext() (pynlpl.formats.folia.PhonContent method),
                                                           strip_accents() (in module pynlpl.textprocessors), 1067
                                                           SubjectivityAnnotation (class in pynlpl.formats.folia),
stricttext() (pynlpl.formats.folia.PosAnnotation method),
                                                           SUBSET (pynlpl.formats.folia.AbstractAnnotationLayer
stricttext() (pynlpl.formats.folia.Predicate method), 583
                                                                     attribute), 75
stricttext() (pynlpl.formats.folia.Quote method), 302
                                                           SUBSET
                                                                        (pynlpl.formats.folia.AbstractElement
                                                                                                                 at-
stricttext() (pynlpl.formats.folia.Reference method), 315
                                                                     tribute), 26
stricttext() (pynlpl.formats.folia.Row method), 328
                                                           SUBSET (pynlpl.formats.folia.AbstractSpanAnnotation
stricttext() (pynlpl.formats.folia.SemanticRole method),
                                                                     attribute), 52
                                                           SUBSET (pynlpl.formats.folia.AbstractStructureElement
stricttext()
               (pynlpl.formats.folia.SemanticRolesLayer
                                                                     attribute), 37
          method), 748
                                                           SUBSET (pynlpl.formats.folia.AbstractTextMarkup at-
stricttext()
                  (pynlpl.formats.folia.SenseAnnotation
                                                                     tribute), 86
          method), 479
                                                           SUBSET (pynlpl.formats.folia.AbstractTokenAnnotation
stricttext() (pynlpl.formats.folia.Sentence method), 342
                                                                     attribute), 63
stricttext() (pynlpl.formats.folia.Sentiment method), 595
                                                           SUBSET (pynlpl.formats.folia.ActorFeature attribute),
                   (pynlpl.formats.folia.SentimentLayer
          method), 713
                                                           SUBSET
                                                                       (pynlpl.formats.folia.Alignment
                                                                                                          attribute),
stricttext() (pynlpl.formats.folia.Statement method), 607
stricttext() (pynlpl.formats.folia.StatementLayer method),
                                                           SUBSET (pynlpl.formats.folia.AlignReference attribute),
stricttext() (pynlpl.formats.folia.SubjectivityAnnotation
                                                           SUBSET (pynlpl.formats.folia.Alternative attribute), 919
                                                                       (pynlpl.formats.folia.AlternativeLayers
          method), 490
                                                           SUBSET
stricttext() (pynlpl.formats.folia.Suggestion
                                                                     tribute), 931
                                                           SUBSET (pynlpl.formats.folia.BegindatetimeFeature at-
                                                                     tribute), 896
stricttext() (pynlpl.formats.folia.SynsetFeature method),
                                                           SUBSET (pynlpl.formats.folia.Cell attribute), 101
stricttext() (pynlpl.formats.folia.SyntacticUnit method),
                                                           SUBSET (pynlpl.formats.folia.Chunk attribute), 518
```

SUBSET (pynlpl.formats.folia.ChunkingLayer attribute), 647 **SUBSET** (pynlpl.formats.folia.CoreferenceChain tribute), 530 (pynlpl.formats.folia.CoreferenceLayer tribute), 658 **SUBSET** (pvnlpl.formats.folia.CoreferenceLink tribute), 764 SUBSET (pynlpl.formats.folia.Correction attribute), 944 SUBSET (pynlpl.formats.folia.Current attribute), 954 SUBSET (pynlpl.formats.folia.Definition attribute), 113 SUBSET (pynlpl.formats.folia.DependenciesLayer attribute), 670 SUBSET (pynlpl.formats.folia.Dependency attribute), **SUBSET** (pynlpl.formats.folia.DependencyDependent attribute), 776 (pynlpl.formats.folia.Description SUBSET SUBSET (pynlpl.formats.folia.Division attribute), 126 (pynlpl.formats.folia.DomainAnnotation **SUBSET** attribute), 429 SUBSET (pynlpl.formats.folia.EnddatetimeFeature attribute), 907 SUBSET (pynlpl.formats.folia.EntitiesLayer attribute), SUBSET (pynlpl.formats.folia.Entity attribute), 553 SUBSET (pynlpl.formats.folia.Entry attribute), 139 SUBSET (pynlpl.formats.folia.ErrorDetection attribute), SUBSET (pynlpl.formats.folia.Event attribute), 152 SUBSET (pynlpl.formats.folia.Example attribute), 165 SUBSET (pynlpl.formats.folia.Feature attribute), 862 SUBSET (pynlpl.formats.folia.Figure attribute), 178 SUBSET (pynlpl.formats.folia.Gap attribute), 191 SUBSET (pynlpl.formats.folia.Head attribute), 202 SUBSET (pynlpl.formats.folia.Headspan attribute), 788 (pynlpl.formats.folia.LangAnnotation **SUBSET** attribute), 451 SUBSET (pynlpl.formats.folia.LemmaAnnotation attribute), 462 SUBSET (pynlpl.formats.folia.Linebreak attribute), 215 SUBSET (pynlpl.formats.folia.List attribute), 228 SUBSET (pynlpl.formats.folia.ListItem attribute), 241 SUBSET (pynlpl.formats.folia.Metric attribute), 1043 SUBSET (pynlpl.formats.folia.New attribute), 976 SUBSET (pynlpl.formats.folia.Note attribute), 254 SUBSET (pynlpl.formats.folia.Observation attribute), 565 **SUBSET** (pynlpl.formats.folia.ObservationLayer tribute), 694 SUBSET (pynlpl.formats.folia.Original attribute), 987 SUBSET (pynlpl.formats.folia.Paragraph attribute), 267

SUBSET (pynlpl.formats.folia.Part attribute), 280

SUBSET (pynlpl.formats.folia.PhonContent attribute), SUBSET (pynlpl.formats.folia.PosAnnotation attribute), SUBSET (pynlpl.formats.folia.Predicate attribute), 576 SUBSET (pynlpl.formats.folia.Quote attribute), 293 SUBSET (pynlpl.formats.folia.Reference attribute), 306 SUBSET (pynlpl.formats.folia.Row attribute), 319 SUBSET (pynlpl.formats.folia.SemanticRole attribute). 623 SUBSET (pynlpl.formats.folia.SemanticRolesLayer attribute), 741 **SUBSET** (pynlpl.formats.folia.SenseAnnotation tribute), 473 SUBSET (pynlpl.formats.folia.Sentence attribute), 333 SUBSET (pynlpl.formats.folia.Sentiment attribute), 588 SUBSET (pynlpl.formats.folia.SentimentLayer attribute), 705 SUBSET (pynlpl.formats.folia.Statement attribute), 599 SUBSET (pynlpl.formats.folia.StatementLayer attribute), SUBSET (pynlpl.formats.folia.SubjectivityAnnotation attribute), 484 SUBSET (pynlpl.formats.folia.Suggestion attribute), 998 SUBSET (pynlpl.formats.folia.SynsetFeature attribute), SUBSET (pynlpl.formats.folia.SyntacticUnit attribute), SUBSET (pynlpl.formats.folia.SyntaxLayer attribute), SUBSET (pynlpl.formats.folia.Table attribute), 347 SUBSET (pynlpl.formats.folia.TableHead attribute), 373 SUBSET (pynlpl.formats.folia.Term attribute), 360 SUBSET (pynlpl.formats.folia.Text attribute), 386 SUBSET (pynlpl.formats.folia.TextContent attribute), SUBSET (pynlpl.formats.folia.TextMarkupCorrection attribute), 840 **SUBSET** (pynlpl.formats.folia.TextMarkupError tribute), 851 **SUBSET** (pynlpl.formats.folia.TextMarkupGap attribute), 808 **SUBSET** (pynlpl.formats.folia.TextMarkupString attribute), 819 **SUBSET** (pynlpl.formats.folia.TextMarkupStyle tribute), 830 SUBSET (pynlpl.formats.folia.TimeSegment attribute), 634 SUBSET (pynlpl.formats.folia.TimingLayer attribute), SUBSET (pynlpl.formats.folia.Whitespace attribute), 399 SUBSET (pynlpl.formats.folia.Word attribute), 413

Suggestion (class in pynlpl.formats.folia), 995

suggestions() (pynlpl.formats.folia.Correction method),

950	text() (pynlpl.formats.folia.DependenciesLayer method),
sum() (pynlpl.statistics.FrequencyList method), 1063	677
swap() (in module pynlpl.textprocessors), 1067	text() (pynlpl.formats.folia.Dependency method), 548
SynsetFeature (class in pynlpl.formats.folia), 871	text() (pynlpl.formats.folia.DependencyDependent
SyntacticUnit (class in pynlpl.formats.folia), 608	method), 783
SyntaxLayer (class in pynlpl.formats.folia), 726	text() (pynlpl.formats.folia.Description method), 1038
Т	text() (pynlpl.formats.folia.Division method), 135
	text() (pynlpl.formats.folia.Document method), 20
Table (class in pynlpl.formats.folia), 344	text() (pynlpl.formats.folia.DomainAnnotation method), 435
TableHead (class in pynlpl.formats.folia), 370	text() (pynlpl.formats.folia.EnddatetimeFeature method),
Taggerdata (class in pynlpl.formats.taggerdata), 1055	914
targetword() (pynlpl.formats.giza.MultiWordAlignment method), 1054	text() (pynlpl.formats.folia.EntitiesLayer method), 689
targetword() (pynlpl.formats.giza.WordAlignment	text() (pynlpl.formats.folia.Entity method), 560
method), 1054	text() (pynlpl.formats.folia.Entry method), 148
targetwords() (pynlpl.formats.giza.MultiWordAlignment	text() (pynlpl.formats.folia.ErrorDetection method), 971
method), 1054	text() (pynlpl.formats.folia.Event method), 161
Term (class in pynlpl.formats.folia), 357	text() (pynlpl.formats.folia.Example method), 174
test() (pynlpl.evaluation.WPSParamSearch method), 10	text() (pynlpl.formats.folia.Feature method), 869
test() (pynlpl.search.AbstractSearchState method), 1060	text() (pynlpl.formats.folia.Figure method), 187
Text (class in pynlpl.formats.folia), 383	text() (pynlpl.formats.folia.Gap method), 198
text() (pynlpl.formats.folia.AbstractAnnotationLayer	text() (pynlpl.formats.folia.Head method), 211
method), 82	text() (pynlpl.formats.folia.Headspan method), 795
text() (pynlpl.formats.folia.AbstractElement method), 32	text() (pynlpl.formats.folia.LangAnnotation method), 457
text() (pynlpl.formats.folia.AbstractSpanAnnotation	text() (pynlpl.formats.folia.LemmaAnnotation method),
method), 59	468
text() (pynlpl.formats.folia.AbstractStructureElement	text() (pynlpl.formats.folia.Linebreak method), 223
method), 46	text() (pynlpl.formats.folia.List method), 237
text() (pynlpl.formats.folia.AbstractTextMarkup method),	text() (pynlpl.formats.folia.ListItem method), 250
93	text() (pynlpl.formats.folia.Metric method), 1049 text() (pynlpl.formats.folia.New method), 982
text() (pynlpl.formats.folia.AbstractTokenAnnotation	text() (pynlpl.formats.folia.New method), 362 text() (pynlpl.formats.folia.Note method), 263
method), 70	text() (pynlpl.formats.folia.Note incurod), 203 text() (pynlpl.formats.folia.Observation method), 572
text() (pynlpl.formats.folia.ActorFeature method), 892 text() (pynlpl.formats.folia.Alignment method), 1016	text() (pynlpl.formats.folia.ObservationLayer method),
text() (pynipi.iorinats.ioria.Ariginnent inethod), 1010 text() (pynipi.formats.folia.AlignReference method),	701
1027	text() (pynlpl.formats.folia.Original method), 993
text() (pynlpl.formats.folia.Alternative method), 927	text() (pynlpl.formats.folia.Paragraph method), 276
text() (pynlpl.formats.folia.AlternativeLayers method),	text() (pynlpl.formats.folia.Part method), 289
938	text() (pynlpl.formats.folia.PhonContent method), 512
text() (pynlpl.formats.folia.BegindatetimeFeature	text() (pynlpl.formats.folia.PosAnnotation method), 446
method), 903	text() (pynlpl.formats.folia.Predicate method), 583
text() (pynlpl.formats.folia.Cell method), 109	text() (pynlpl.formats.folia.Quote method), 302
text() (pynlpl.formats.folia.Chunk method), 525	text() (pynlpl.formats.folia.Reference method), 315
text() (pynlpl.formats.folia.ChunkingLayer method), 654	text() (pynlpl.formats.folia.Row method), 328
text() (pynlpl.formats.folia.CoreferenceChain method),	text() (pynlpl.formats.folia.SemanticRole method), 630
537	text() (pynlpl.formats.folia.SemanticRolesLayer method),
text() (pynlpl.formats.folia.CoreferenceLayer method),	748
666	text() (pynlpl.formats.folia.SenseAnnotation method),
text() (pynlpl.formats.folia.CoreferenceLink method),	text() (pynlpl.formats.folia.Sentence method), 342
772	text() (pynlpl.formats.folia.Sentiment method), 542 text() (pynlpl.formats.folia.Sentiment method), 595
text() (pynlpl.formats.folia.Correction method), 950	text() (pynlpl.formats.folia.Sentiment Inethod), 393 text() (pynlpl.formats.folia.SentimentLayer method), 713
text() (pynlpl.formats.folia.Current method), 960 text() (pynlpl.formats.folia.Definition method), 122	text() (pynlpl.formats.folia.Statement method), 607
text() (pympi.ioimats.iona.peminton method), 122	text() (pynlpl.formats.folia.StatementLayer method), 724

tout() (numbel formats folio Subjectivity Apparation	TEVTCONTAINED (number formats folio CoreferenceChain
text() (pynlpl.formats.folia.SubjectivityAnnotation method), 490	TEXTCONTAINER (pynlpl.formats.folia.CoreferenceChain attribute), 530
	TEXTCONTAINER (pynlpl.formats.folia.CoreferenceLayer
text() (pynlpl.formats.folia.SynsetFeature method), 880	attribute), 658
	TEXTCONTAINER (pynlpl.formats.folia.CoreferenceLink
text() (pynlpl.formats.folia.SyntaxLayer method), 736	attribute), 764
	TEXTCONTAINER (pynlpl.formats.folia.Correction at-
text() (pynlpl.formats.folia.TableHead method), 381	tribute), 944
	TEXTCONTAINER (pynlpl.formats.folia.Current at-
text() (pynlpl.formats.folia.Text method), 394 text() (pynlpl.formats.folia.TextContent method), 502	tribute), 954 TEXTCONTAINER (pynlpl.formats.folia.Definition at-
text() (pynlpl.formats.folia.TextMarkupCorrection	tribute), 113
	TEXTCONTAINER (pynlpl.formats.folia.DependenciesLayer
text() (pynlpl.formats.folia.TextMarkupError method),	attribute), 670
	TEXTCONTAINER (pynlpl.formats.folia.Dependency
text() (pynlpl.formats.folia.TextMarkupGap method), 815	attribute), 541
text() (pynlpl.formats.folia.TextMarkupString method),	TEXTCONTAINER (pynlpl.formats.folia.DependencyDependent
826	attribute), 776
	TEXTCONTAINER (pynlpl.formats.folia.Description at-
836	tribute), 1032
text() (pynlpl.formats.folia.TimeSegment method), 642 text() (pynlpl.formats.folia.TimingLayer method), 760	TEXTCONTAINER (pynlpl.formats.folia.Division at-
	tribute), 126 TEXTCONTAINER (pynlpl.formats.folia.DomainAnnotation
text() (pynlpl.formats.folia.Word method), 422	attribute), 429
	GHEXIGEONTAINER (pynlpl.formats.folia.EnddatetimeFeature
attribute), 75	attribute), 907
TEXTCONTAINER (pynlpl.formats.folia.AbstractElement	
attribute), 26	attribute), 682
TEXTCONTAINER (pynlpl.formats.folia.AbstractSpanAnn	** *
attribute), 52	tribute), 553
TEXTCONTAINER (pynlpl.formats.folia.AbstractStructure	** *
attribute), 37 TEXTCONTAINER (pynlpl.formats.folia.AbstractTextMark	tribute), 139
attribute), 86	attribute), 965
TEXTCONTAINER (pynlpl.formats.folia.AbstractTokenAn	
attribute), 63	tribute), 152
	TEXTCONTAINER (pynlpl.formats.folia.Example at-
attribute), 885	tribute), 165
47 1	TEXTCONTAINER (pynlpl.formats.folia.Feature at-
tribute), 1009	tribute), 862
TEXTCONTAINER (pynlpl.formats.folia.AlignReference	
attribute), 1020 TEXTCONTAINER (pynlpl.formats.folia.Alternative at-	tribute), 178 TEXTCONTAINER (pynlpl.formats.folia.Gap attribute),
tribute), 919	191
TEXTCONTAINER (pynlpl.formats.folia.AlternativeLayers	
attribute), 931	tribute), 203
TEXTCONTAINER (pynlpl.formats.folia.BegindatetimeFea	
attribute), 896	tribute), 788
	TEXTCONTAINER (pynlpl.formats.folia.LangAnnotation
101	attribute), 451
2.7 2	TEXTCONTAINER (pynlpl.formats.folia.LemmaAnnotation
tribute), 518	attribute), 462
TEXTCONTAINER (pynlpl.formats.folia.ChunkingLayer attribute), 647	tribute), 215
auroucj, 04/	110utc), 213

attribute), 874

TEXTCONTAINER (pynlpl.formats.folia.List attribute), TEXTCONTAINER (pynlpl.formats.folia.SyntacticUnit attribute), 611 228 TEXTCONTAINER (pynlpl.formats.folia.ListItem at-TEXTCONTAINER (pynlpl.formats.folia.SyntaxLayer tribute), 241 attribute), 729 (pynlpl.formats.folia.Metric (pynlpl.formats.folia.Table TEXTCONTAINER **TEXTCONTAINER** tribute), 1043 tribute), 347 TEXTCONTAINER (pynlpl.formats.folia.New attribute), TEXTCONTAINER (pynlpl.formats.folia.TableHead attribute), 373 (pynlpl.formats.folia.Term TEXTCONTAINER (pynlpl.formats.folia.Note attribute), **TEXTCONTAINER** at-254 tribute), 360 TEXTCONTAINER (pynlpl.formats.folia.Observation TEXTCONTAINER (pynlpl.formats.folia.Text attribute), attribute), 565 386 TEXTCONTAINER (pynlpl.formats.folia.ObservationLayerTEXTCONTAINER (pynlpl.formats.folia.TextContent attribute), 694 attribute), 495 **TEXTCONTAINER** (pynlpl.formats.folia.Original TEXTCONTAINER (pynlpl.formats.folia.TextMarkupCorrection attribute), 987 attribute), 840 TEXTCONTAINER (pynlpl.formats.folia.Paragraph at-TEXTCONTAINER (pynlpl.formats.folia.TextMarkupError tribute), 267 attribute), 851 TEXTCONTAINER (pynlpl.formats.folia.Part attribute), TEXTCONTAINER (pynlpl.formats.folia.TextMarkupGap attribute), 808 TEXTCONTAINER (pynlpl.formats.folia.PhonContent TEXTCONTAINER (pynlpl.formats.folia.TextMarkupString attribute), 506 attribute), 819 TEXTCONTAINER (pynlpl.formats.folia.PosAnnotation TEXTCONTAINER (pynlpl.formats.folia.TextMarkupStyle attribute), 440 attribute), 830 TEXTCONTAINER (pynlpl.formats.folia.Predicate at-TEXTCONTAINER (pynlpl.formats.folia.TimeSegment tribute), 576 attribute), 634 TEXTCONTAINER (pynlpl.formats.folia.Quote TEXTCONTAINER (pynlpl.formats.folia.TimingLayer atattribute), 752 tribute), 293 TEXTCONTAINER (pynlpl.formats.folia.Reference at-TEXTCONTAINER (pynlpl.formats.folia.Whitespace attribute), 306 tribute), 399 TEXTCONTAINER (pynlpl.formats.folia.Row attribute), **TEXTCONTAINER** (pynlpl.formats.folia.Word attribute), 413 TEXTCONTAINER (pynlpl.formats.folia.SemanticRole TextContent (class in pynlpl.formats.folia), 492 attribute), 623  $textcontent()\ (pynlpl.formats.folia. Abstract Annotation Layer$ TEXTCONTAINER (pynlpl.formats.folia.SemanticRolesLayer method), 83 (pynlpl.formats.folia.AbstractElement attribute), 741 textcontent() TEXTCONTAINER (pynlpl.formats.folia.SenseAnnotation method), 33 attribute), 473 textcontent() (pynlpl.formats.folia.AbstractSpanAnnotation TEXTCONTAINER (pynlpl.formats.folia.Sentence atmethod), 60 tribute), 333 textcontent() (pynlpl.formats.folia.AbstractStructureElement TEXTCONTAINER (pynlpl.formats.folia.Sentiment atmethod), 46 tribute), 588 textcontent() (pynlpl.formats.folia.AbstractTextMarkup TEXTCONTAINER (pynlpl.formats.folia.SentimentLayer method), 93 attribute), 705 textcontent() (pynlpl.formats.folia.AbstractTokenAnnotation TEXTCONTAINER (pynlpl.formats.folia.Statement atmethod), 71 textcontent() (pynlpl.formats.folia.ActorFeature method), tribute), 599 TEXTCONTAINER (pynlpl.formats.folia.StatementLayer attribute), 717 textcontent() (pynlpl.formats.folia.Alignment method), TEXTCONTAINER (pynlpl.formats.folia.SubjectivityAnnotation 1016 (pynlpl.formats.folia.AlignReference attribute), 484 textcontent() TEXTCONTAINER (pynlpl.formats.folia.Suggestion atmethod), 1027 tribute), 998 textcontent() (pynlpl.formats.folia.Alternative method), TEXTCONTAINER (pynlpl.formats.folia.SynsetFeature 927

1174 Index

textcontent()

(pynlpl.formats.folia.AlternativeLayers

method), 939 textcontent() (pynlpl.formats.folia.BegindatetimeFeature method), 903 textcontent() (pynlpl.formats.folia.Cell method), 109 textcontent() (pynlpl.formats.folia.Chunk method), 526 textcontent() (pynlpl.formats.folia.ChunkingLayer method), 654 (pynlpl.formats.folia.CoreferenceChain textcontent() method), 537 (pynlpl.formats.folia.CoreferenceLayer textcontent() method), 666 (pynlpl.formats.folia.CoreferenceLink textcontent() method), 772 textcontent() (pynlpl.formats.folia.Correction method), 950 textcontent() (pynlpl.formats.folia.Current method), 961 textcontent() (pynlpl.formats.folia.Definition method), 122 (pynlpl.formats.folia.DependenciesLayer textcontent() method), 678 textcontent() (pynlpl.formats.folia.Dependency method), textcontent() (pynlpl.formats.folia.DependencyDependent textcontent() method), 784 textcontent() (pynlpl.formats.folia.Description method), textcontent() (pynlpl.formats.folia.Division method), 135 (pynlpl.formats.folia.DomainAnnotation textcontent() method), 436 (pynlpl.formats.folia.EnddatetimeFeature textcontent() method), 914 textcontent() (pynlpl.formats.folia.EntitiesLayer method), textcontent() (pynlpl.formats.folia.Entity method), 561 textcontent() (pynlpl.formats.folia.Entry method), 148 (pynlpl.formats.folia.ErrorDetection textcontent() method), 972 textcontent() (pynlpl.formats.folia.Event method), 161 textcontent() (pynlpl.formats.folia.Example method), 174 textcontent() (pynlpl.formats.folia.Feature method), 870 textcontent() (pynlpl.formats.folia.Figure method), 187 textcontent() (pynlpl.formats.folia.Gap method), 198 textcontent() (pynlpl.formats.folia.Head method), 211 textcontent() (pynlpl.formats.folia.Headspan method), 796 (pynlpl.formats.folia.LangAnnotation textcontent() method), 458 (pynlpl.formats.folia.LemmaAnnotation textcontent() method), 469 textcontent() (pynlpl.formats.folia.Linebreak method), 224 textcontent() (pynlpl.formats.folia.List method), 237 textcontent() (pynlpl.formats.folia.ListItem method), 250

textcontent() (pynlpl.formats.folia.Metric method), 1050

- textcontent() (pynlpl.formats.folia.New method), 983 textcontent() (pynlpl.formats.folia.Note method), 263 textcontent() (pynlpl.formats.folia.Observation method), 572
- textcontent() (pynlpl.formats.folia.ObservationLayer method), 702
- textcontent() (pynlpl.formats.folia.Original method), 994 textcontent() (pynlpl.formats.folia.Paragraph method), 276
- textcontent() (pynlpl.formats.folia.Part method), 289 textcontent() (pynlpl.formats.folia.PhonContent method), 512
- textcontent() (pynlpl.formats.folia.PosAnnotation method), 447
- textcontent() (pynlpl.formats.folia.Predicate method), 584
- textcontent() (pynlpl.formats.folia.Quote method), 302 textcontent() (pynlpl.formats.folia.Reference method), 315
- textcontent() (pynlpl.formats.folia.Row method), 328 textcontent() (pynlpl.formats.folia.SemanticRole method), 631
- textcontent() (pynlpl.formats.folia.SemanticRolesLayer method), 749
- textcontent() (pynlpl.formats.folia.SenseAnnotation method), 480
- textcontent() (pynlpl.formats.folia.Sentence method), 343 textcontent() (pynlpl.formats.folia.Sentiment method), 596
- textcontent() (pynlpl.formats.folia.SentimentLayer method), 713
- textcontent() (pynlpl.formats.folia.Statement method), 607
- textcontent() (pynlpl.formats.folia.StatementLayer method), 725
- $textcontent () \ (pynlpl. formats. folia. Subjectivity Annotation \\ method), \ 491$
- textcontent() (pynlpl.formats.folia.Suggestion method), 1005
- textcontent() (pynlpl.formats.folia.SynsetFeature method), 881
- textcontent() (pynlpl.formats.folia.SyntacticUnit method), 619
- textcontent() (pynlpl.formats.folia.SyntaxLayer method), 737
- textcontent() (pynlpl.formats.folia.Table method), 356 textcontent() (pynlpl.formats.folia.TableHead method), 382
- textcontent() (pynlpl.formats.folia.Term method), 369 textcontent() (pynlpl.formats.folia.Text method), 395 textcontent() (pynlpl.formats.folia.TextContent method), 502
- textcontent() (pynlpl.formats.folia.TextMarkupCorrection method), 848

tribute), 954 textcontent() (pynlpl.formats.folia.TextMarkupError method), 858 TEXTDELIMITER (pynlpl.formats.folia.Definition attextcontent() (pynlpl.formats.folia.TextMarkupGap tribute), 113 TEXTDELIMITER (pynlpl.formats.folia.DependenciesLayer method), 816 textcontent() (pynlpl.formats.folia.TextMarkupString attribute), 670 method), 826 TEXTDELIMITER (pynlpl.formats.folia.Dependency at-(pynlpl.formats.folia.TextMarkupStyle textcontent() tribute), 541 method), 837 TEXTDELIMITER (pynlpl.formats.folia.DependencyDependent (pynlpl.formats.folia.TimeSegment textcontent() attribute), 776 method), 642 TEXTDELIMITER (pynlpl.formats.folia.Description attextcontent() (pynlpl.formats.folia.TimingLayer method), tribute), 1032 **TEXTDELIMITER** (pynlpl.formats.folia.Division textcontent() (pynlpl.formats.folia.Whitespace method), attribute), 126 TEXTDELIMITER (pynlpl.formats.folia.DomainAnnotation 408 textcontent() (pynlpl.formats.folia.Word method), 423 attribute), 429 TEXTDELIMITER (pynlpl.formats.folia.AbstractAnnotation EXTEDELIMITER (pynlpl.formats.folia.EnddatetimeFeature attribute), 75 attribute), 907 TEXTDELIMITER (pynlpl.formats.folia.AbstractElement TEXTDELIMITER (pynlpl.formats.folia.EntitiesLayer attribute), 26 attribute), 682 TEXTDELIMITER (pynlpl.formats.folia.AbstractSpanAnnoFarXofiDELIMITER (pynlpl.formats.folia.Entity attribute), 52 tribute), 553 TEXTDELIMITER (pynlpl.formats.folia.AbstractStructureHELIMITER (pynlpl.formats.folia.Entry attribute), attribute), 37 139 TEXTDELIMITER (pynlpl.formats.folia.AbstractTextMarktijEXTDELIMITER (pynlpl.formats.folia.ErrorDetection attribute), 86 attribute), 965 TEXTDELIMITER (pynlpl.formats.folia.AbstractTokenAnnDEXTODELIMITER (pynlpl.formats.folia.Event attribute), 63 tribute), 152 TEXTDELIMITER (pynlpl.formats.folia.ActorFeature TEXTDELIMITER (pynlpl.formats.folia.Example attribute), 885 attribute), 165 (pynlpl.formats.folia.Feature TEXTDELIMITER (pynlpl.formats.folia.Alignment at-**TEXTDELIMITER** tribute), 1009 tribute), 862 (pynlpl.formats.folia.Figure TEXTDELIMITER (pynlpl.formats.folia.AlignReference TEXTDELIMITER atattribute), 1020 tribute), 178 TEXTDELIMITER (pynlpl.formats.folia.Alternative at- TEXTDELIMITER (pynlpl.formats.folia.Gap attribute), tribute), 919 191 TEXTDELIMITER (pynlpl.formats.folia.AlternativeLayers TEXTDELIMITER (pynlpl.formats.folia.Head attribute), attribute), 931 203 TEXTDELIMITER (pynlpl.formats.folia.BegindatetimeFeatilieXTDELIMITER (pynlpl.formats.folia.Headspan atattribute), 896 tribute), 788 TEXTDELIMITER (pynlpl.formats.folia.Cell attribute), TEXTDELIMITER (pynlpl.formats.folia.LangAnnotation attribute), 451 TEXTDELIMITER (pynlpl.formats.folia.Chunk TEXTDELIMITER (pynlpl.formats.folia.LemmaAnnotation tribute), 518 attribute), 462 TEXTDELIMITER (pynlpl.formats.folia.ChunkingLayer TEXTDELIMITER (pynlpl.formats.folia.Linebreak atattribute), 647 tribute), 215 TEXTDELIMITER (pynlpl.formats.folia.CoreferenceChainTEXTDELIMITER (pynlpl.formats.folia.List attribute), attribute), 530 228 TEXTDELIMITER (pynlpl.formats.folia.CoreferenceLayerTEXTDELIMITER (pynlpl.formats.folia.ListItem attribute), 658 attribute), 241 TEXTDELIMITER (pynlpl.formats.folia.CoreferenceLink TEXTDELIMITER (pynlpl.formats.folia.Metric attribute), 764 tribute), 1043 TEXTDELIMITER (pynlpl.formats.folia.Correction at-TEXTDELIMITER (pynlpl.formats.folia.New attribute), tribute), 944 TEXTDELIMITER (pynlpl.formats.folia.Current at TEXTDELIMITER (pynlpl.formats.folia.Note attribute),

254	360
TEXTDELIMITER (pynlpl.formats.folia.Observation at-	TEXTDELIMITER (pynlpl.formats.folia.Text attribute),
tribute), 565	386
	er TEXTDELIMITER (pynlpl.formats.folia.TextContent at-
attribute), 694	tribute), 495
TEXTDELIMITER (pynlpl.formats.folia.Original attribute), 987	TEXTDELIMITER (pynlpl.formats.folia.TextMarkupCorrection attribute), 841
TEXTDELIMITER (pynlpl.formats.folia.Paragraph at-	TEXTDELIMITER (pynlpl.formats.folia.TextMarkupError
tribute), 267	attribute), 851
TEXTDELIMITER (pynlpl.formats.folia.Part attribute), 280	TEXTDELIMITER (pynlpl.formats.folia.TextMarkupGap attribute), 809
TEXTDELIMITER (pynlpl.formats.folia.PhonContent attribute), 506	TEXTDELIMITER (pynlpl.formats.folia.TextMarkupString attribute), 819
TEXTDELIMITER (pynlpl.formats.folia.PosAnnotation	TEXTDELIMITER (pynlpl.formats.folia.TextMarkupStyle
attribute), 440	attribute), 830
TEXTDELIMITER (pynlpl.formats.folia.Predicate at-	TEXTDELIMITER (pynlpl.formats.folia.TimeSegment
tribute), 576	attribute), 634
TEXTDELIMITER (pynlpl.formats.folia.Quote attribute), 293	TEXTDELIMITER (pynlpl.formats.folia.TimingLayer attribute), 753
TEXTDELIMITER (pynlpl.formats.folia.Reference at-	TEXTDELIMITER (pynlpl.formats.folia.Whitespace at-
tribute), 306	tribute), 399
TEXTDELIMITER (pynlpl.formats.folia.Row attribute),	TEXTDELIMITER (pynlpl.formats.folia.Word attribute),
319	413
TEXTDELIMITER (pynlpl.formats.folia.SemanticRole	TextMarkupCorrection (class in pynlpl.formats.folia),
attribute), 623	838
TEXTDELIMITER (pynlpl.formats.folia.SemanticRolesL attribute), 741	ayeextMarkupError (class in pynlpl.formats.folia), 849 TextMarkupGap (class in pynlpl.formats.folia), 806
TEXTDELIMITER (pynlpl.formats.folia.SenseAnnotation	
attribute), 473	TextMarkupStyle (class in pynlpl.formats.folia), 827
TEXTDELIMITER (pynlpl.formats.folia.Sentence	textvalidation() (pynlpl.formats.folia.AbstractAnnotationLayer
attribute), 333	method), 83
TEXTDELIMITER (pynlpl.formats.folia.Sentiment at-	textvalidation() (pynlpl.formats.folia.AbstractElement
tribute), 588	method), 33
TEXTDELIMITER (pynlpl.formats.folia.SentimentLayer attribute), 706	textvalidation() (pynlpl.formats.folia.AbstractSpanAnnotation method), 60
TEXTDELIMITER (pynlpl.formats.folia.Statement at-	textvalidation() (pynlpl.formats.folia.AbstractStructureElement
tribute), 599	method), 47
	textvalidation() (pynlpl.formats.folia.AbstractTextMarkup
attribute), 717	method), 94
2.7 2	otation() (pynlpl.formats.folia.AbstractTokenAnnotation
attribute), 484	method), 71
TEXTDELIMITER (pynlpl.formats.folia.Suggestion at-	textvalidation() (pynlpl.formats.folia.ActorFeature
tribute), 998 TEXTDELIMITER (pynlpl.formats.folia.SynsetFeature	method), 893 textvalidation() (pynlpl.formats.folia.Alignment method),
attribute), 874	1017
TEXTDELIMITER (pynlpl.formats.folia.SyntacticUnit	textvalidation() (pynlpl.formats.folia.AlignReference
attribute), 611	method), 1028
TEXTDELIMITER (pynlpl.formats.folia.SyntaxLayer	textvalidation() (pynlpl.formats.folia.Alternative
attribute), 729	method), 928
TEXTDELIMITER (pynlpl.formats.folia.Table attribute),	textvalidation() (pynlpl.formats.folia.AlternativeLayers
347 TEXTDELIMITER (pynlpl.formats.folia.TableHead at-	method), 939 textvalidation() (pynlpl.formats.folia.BegindatetimeFeature
tribute), 373	warvandauon() (pympi.ioimais.iona.begindateumereature
	method), 904
TEXTDELIMITER (pynlpl.formats.folia.Term attribute),	method), 904 textvalidation() (pynlpl.formats.folia.Cell method), 110

- textvalidation() (pynlpl.formats.folia.Chunk method), 526 (pynlpl.formats.folia.ChunkingLayer textvalidation() method), 655
- textvalidation() (pynlpl.formats.folia.CoreferenceChain method), 538
- textvalidation() (pynlpl.formats.folia.CoreferenceLayer method), 667
- textvalidation() (pynlpl.formats.folia.CoreferenceLink method), 773
- textvalidation() (pynlpl.formats.folia.Correction method),
- textvalidation() (pynlpl.formats.folia.Current method),
- textvalidation() (pynlpl.formats.folia.Definition method),
- textvalidation() (pynlpl.formats.folia.DependenciesLayer method), 678
- (pynlpl.formats.folia.Dependency textvalidation() method), 549
- textvalidation() (pynlpl.formats.folia.DependencyDependentextvalidation() (pynlpl.formats.folia.Quote method), 303 method), 784
- textvalidation() (pynlpl.formats.folia.Description method), 1039
- textvalidation() (pynlpl.formats.folia.Division method), 136
- textvalidation() (pynlpl.formats.folia.DomainAnnotation method), 436
- textvalidation() (pynlpl.formats.folia.EnddatetimeFeature method), 915
- textvalidation() (pynlpl.formats.folia.EntitiesLayer method), 690
- textvalidation() (pynlpl.formats.folia.Entity method), 561 textvalidation() (pynlpl.formats.folia.Entry method), 149
- (pynlpl.formats.folia.ErrorDetection textvalidation() method), 972
- textvalidation() (pynlpl.formats.folia.Event method), 162 textvalidation() (pynlpl.formats.folia.Example method), 175
- textvalidation() (pynlpl.formats.folia.Feature method), 870
- textvalidation() (pynlpl.formats.folia.Figure method), 188
- textvalidation() (pynlpl.formats.folia.Gap method), 199
- textvalidation() (pynlpl.formats.folia.Head method), 212 textvalidation() (pynlpl.formats.folia.Headspan method),
- 796 textvalidation() (pynlpl.formats.folia.LangAnnotation
- method), 458 textvalidation() (pynlpl.formats.folia.LemmaAnnotation
- method), 469 textvalidation() (pynlpl.formats.folia.Linebreak method),
- 224
- textvalidation() (pynlpl.formats.folia.List method), 238 textvalidation() (pynlpl.formats.folia.ListItem method), 251

- textvalidation() (pynlpl.formats.folia.Metric method), 1050
- textvalidation() (pynlpl.formats.folia.New method), 983 textvalidation() (pynlpl.formats.folia.Note method), 264
- textvalidation() (pynlpl.formats.folia.Observation method), 573
- textvalidation() (pynlpl.formats.folia.ObservationLayer method), 702
- textvalidation() (pynlpl.formats.folia.Original method),
- textvalidation() (pynlpl.formats.folia.Paragraph method),
- textvalidation() (pynlpl.formats.folia.Part method), 290 (pynlpl.formats.folia.PhonContent textvalidation()
- method), 513 textvalidation() (pynlpl.formats.folia.PosAnnotation
- method), 447 textvalidation() (pynlpl.formats.folia.Predicate method),
- textvalidation() (pynlpl.formats.folia.Reference method),
- textvalidation() (pynlpl.formats.folia.Row method), 329 textvalidation() (pynlpl.formats.folia.SemanticRole method), 631
- textvalidation() (pynlpl.formats.folia.SemanticRolesLayer method), 749
- textvalidation() (pynlpl.formats.folia.SenseAnnotation method), 480
- textvalidation() (pynlpl.formats.folia.Sentence method),
- textvalidation() (pynlpl.formats.folia.Sentiment method), 596
- textvalidation() (pynlpl.formats.folia.SentimentLayer method), 714
- textvalidation() (pynlpl.formats.folia.Statement method), 608
- textvalidation() (pynlpl.formats.folia.StatementLayer method), 725
- textvalidation() (pynlpl.formats.folia.SubjectivityAnnotation method), 491
- textvalidation() (pynlpl.formats.folia.Suggestion method), 1005
- textvalidation() (pynlpl.formats.folia.SynsetFeature method), 881
- textvalidation() (pynlpl.formats.folia.SyntacticUnit method), 619
- textvalidation() (pynlpl.formats.folia.SyntaxLayer method), 737
- textvalidation() (pynlpl.formats.folia.Table method), 356 textvalidation() (pynlpl.formats.folia.TableHead method),
- textvalidation() (pynlpl.formats.folia.Term method), 369 textvalidation() (pynlpl.formats.folia.Text method), 395

(pynlpl.formats.folia.TextContent toktext() textvalidation() (pynlpl.formats.folia.CoreferenceLayer method), 502 method), 667 textvalidation() (pynlpl.formats.folia.TextMarkupCorrectiontoktext() (pynlpl.formats.folia.CoreferenceLink method), method), 848 toktext() (pynlpl.formats.folia.Correction method), 950 textvalidation() (pynlpl.formats.folia.TextMarkupError method), 859 toktext() (pynlpl.formats.folia.Current method), 961 (pynlpl.formats.folia.TextMarkupGap textvalidation() toktext() (pynlpl.formats.folia.Definition method), 123 (pynlpl.formats.folia.DependenciesLayer method), 816 toktext() textvalidation() (pynlpl.formats.folia.TextMarkupString method), 679 method), 827 toktext() (pynlpl.formats.folia.Dependency method), 550 textvalidation() (pynlpl.formats.folia.TextMarkupStyle toktext() (pynlpl.formats.folia.DependencyDependent method), 837 method), 784 (pynlpl.formats.folia.TimeSegment toktext() (pynlpl.formats.folia.Description method), 1039 textvalidation() method), 643 toktext() (pynlpl.formats.folia.Division method), 136 (pynlpl.formats.folia.DomainAnnotation textvalidation() (pynlpl.formats.folia.TimingLayer toktext() method), 761 method), 436 textvalidation() (pynlpl.formats.folia.Whitespace toktext() (pynlpl.formats.folia.EnddatetimeFeature method), 915 method), 408 (pynlpl.formats.folia.EntitiesLayer method), textvalidation() (pynlpl.formats.folia.Word method), 423 toktext() TimblOutput (class in pynlpl.formats.timbl), 1055 TimeSegment (class in pynlpl.formats.folia), 632 toktext() (pynlpl.formats.folia.Entity method), 561 TimingLayer (class in pynlpl.formats.folia), 750 toktext() (pynlpl.formats.folia.Entry method), 149 title() (pynlpl.formats.folia.Document method), 20 toktext() (pynlpl.formats.folia.ErrorDetection method), tokenise() (in module pynlpl.textprocessors), 1067 tokenize() (in module pynlpl.textprocessors), 1067 toktext() (pynlpl.formats.folia.Event method), 162 Tokenizer (class in pynlpl.textprocessors), 1066 toktext() (pynlpl.formats.folia.Example method), 175 tokens() (pynlpl.statistics.FrequencyList method), 1063 toktext() (pynlpl.formats.folia.Feature method), 870 toktext() (pynlpl.formats.folia.AbstractAnnotationLayer toktext() (pynlpl.formats.folia.Figure method), 188 toktext() (pynlpl.formats.folia.Gap method), 199 method), 83 toktext() (pynlpl.formats.folia.AbstractElement method), toktext() (pynlpl.formats.folia.Head method), 212 toktext() (pynlpl.formats.folia.Headspan method), 796 toktext() (pynlpl.formats.folia.AbstractSpanAnnotation toktext() (pynlpl.formats.folia.LangAnnotation method), method), 60 458 toktext() (pynlpl.formats.folia.AbstractStructureElement toktext() (pynlpl.formats.folia.LemmaAnnotation method), 469 method), 47 (pynlpl.formats.folia.AbstractTextMarkup toktext() toktext() (pynlpl.formats.folia.Linebreak method), 225 method), 94 toktext() (pynlpl.formats.folia.List method), 238 toktext() (pynlpl.formats.folia.AbstractTokenAnnotation toktext() (pynlpl.formats.folia.ListItem method), 251 toktext() (pynlpl.formats.folia.Metric method), 1050 method), 71 toktext() (pynlpl.formats.folia.ActorFeature method), 893 toktext() (pynlpl.formats.folia.New method), 983 toktext() (pynlpl.formats.folia.Alignment method), 1017 toktext() (pynlpl.formats.folia.Note method), 264 toktext() (pynlpl.formats.folia.AlignReference method), toktext() (pynlpl.formats.folia.Observation method), 573 toktext() (pynlpl.formats.folia.ObservationLayer toktext() (pynlpl.formats.folia.Alternative method), 928 method), 702 (pynlpl.formats.folia.AlternativeLayers toktext() (pynlpl.formats.folia.Original method), 994 toktext() method), 939 toktext() (pynlpl.formats.folia.Paragraph method), 277 (pynlpl.formats.folia.BegindatetimeFeature toktext() (pynlpl.formats.folia.Part method), 290 toktext() toktext() (pynlpl.formats.folia.PhonContent method), 513 method), 904 toktext() (pynlpl.formats.folia.Cell method), 110 toktext() (pynlpl.formats.folia.PosAnnotation method), toktext() (pynlpl.formats.folia.Chunk method), 526 toktext() (pynlpl.formats.folia.ChunkingLayer method), toktext() (pynlpl.formats.folia.Predicate method), 584 toktext() (pynlpl.formats.folia.Quote method), 303 (pynlpl.formats.folia.CoreferenceChain toktext() (pynlpl.formats.folia.Reference method), 316 toktext() method), 538 toktext() (pynlpl.formats.folia.Row method), 329

- toktext() (pynlpl.formats.folia.SemanticRole method), 631 (pynlpl.formats.folia.SemanticRolesLayer toktext() method), 749 toktext() (pynlpl.formats.folia.SenseAnnotation method), toktext() (pynlpl.formats.folia.Sentence method), 343 toktext() (pynlpl.formats.folia.Sentiment method), 596 toktext() (pynlpl.formats.folia.SentimentLayer method), toktext() (pynlpl.formats.folia.Statement method), 608 toktext() (pynlpl.formats.folia.StatementLayer method), (pynlpl.formats.folia.SubjectivityAnnotation toktext() method), 491 toktext() (pynlpl.formats.folia.Suggestion method), 1005 toktext() (pynlpl.formats.folia.SynsetFeature method), toktext() (pynlpl.formats.folia.SyntacticUnit method), toktext() (pynlpl.formats.folia.SyntaxLayer method), 737 toktext() (pynlpl.formats.folia.Table method), 356 toktext() (pynlpl.formats.folia.TableHead method), 382 toktext() (pynlpl.formats.folia.Term method), 369 toktext() (pynlpl.formats.folia.Text method), 396 toktext() (pynlpl.formats.folia.TextContent method), 502 toktext() (pynlpl.formats.folia.TextMarkupCorrection method), 848 toktext() (pynlpl.formats.folia.TextMarkupError method), toktext() (pynlpl.formats.folia.TextMarkupGap method), 816 (pynlpl.formats.folia.TextMarkupString toktext() method), 827 toktext() (pynlpl.formats.folia.TextMarkupStyle method), toktext() (pynlpl.formats.folia.TimeSegment method), toktext() (pynlpl.formats.folia.TimingLayer method), 761 toktext() (pynlpl.formats.folia.Whitespace method), 409 toktext() (pynlpl.formats.folia.Word method), 424 tp\_rate() (pynlpl.evaluation.ClassEvaluation method), 10 traversal() (pynlpl.search.AbstractSearch method), 1059 traversal() (pynlpl.search.IterativeDeepening method), 1060 traversalsize() (pynlpl.search.AbstractSearch method), 1059 traversalsize() (pynlpl.search.IterativeDeepening method), 1060 Tree (class in pynlpl.datatypes), 6 Trie (class in pynlpl.datatypes), 6 typetokenratio() (pynlpl.statistics.FrequencyList method),
- U u() (in module pynlpl.common), 3 unalias() (pynlpl.formats.folia.Document method), 20 updatetext() (pynlpl.formats.folia.AbstractAnnotationLayer method), 83 updatetext() (pynlpl.formats.folia.AbstractElement method), 34 updatetext() (pynlpl.formats.folia.AbstractSpanAnnotation method), 60 updatetext() (pynlpl.formats.folia.AbstractStructureElement method), 47 updatetext() (pynlpl.formats.folia.AbstractTextMarkup method), 94 updatetext() (pynlpl.formats.folia.AbstractTokenAnnotation method), 71 updatetext() (pynlpl.formats.folia.ActorFeature method), updatetext() (pynlpl.formats.folia.Alignment method), 1017 (pynlpl.formats.folia.AlignReference updatetext() method), 1028 updatetext() (pynlpl.formats.folia.Alternative method), 928 (pynlpl.formats.folia.AlternativeLayers updatetext() method), 939 updatetext() (pynlpl.formats.folia.BegindatetimeFeature method), 904 updatetext() (pynlpl.formats.folia.Cell method), 110 updatetext() (pynlpl.formats.folia.Chunk method), 526 (pynlpl.formats.folia.ChunkingLayer updatetext() method), 655 (pynlpl.formats.folia.CoreferenceChain updatetext() method), 538 (pynlpl.formats.folia.CoreferenceLayer updatetext() method), 667 (pynlpl.formats.folia.CoreferenceLink updatetext() method), 773 updatetext() (pynlpl.formats.folia.Correction method), updatetext() (pynlpl.formats.folia.Current method), 961 updatetext() (pynlpl.formats.folia.Definition method), 123 updatetext() (pynlpl.formats.folia.DependenciesLayer method), 679 updatetext() (pynlpl.formats.folia.Dependency method), 550 updatetext() (pynlpl.formats.folia.DependencyDependent method), 784 updatetext() (pynlpl.formats.folia.Description method),
- updatetext() (pynlpl.formats.folia.EnddatetimeFeature

updatetext() (pynlpl.formats.folia.Division method), 136

(pynlpl.formats.folia.DomainAnnotation

1180 Index

1039

method), 436

updatetext()

method), 915	608
updatetext() (pynlpl.formats.folia.EntitiesLayer method), 690	updatetext() (pynlpl.formats.folia.StatementLayer method), 726
updatetext() (pynlpl.formats.folia.Entity method), 561 updatetext() (pynlpl.formats.folia.Entry method), 149	updatetext() (pynlpl.formats.folia.SubjectivityAnnotation method), 491
updatetext() (pynlpl.formats.folia.ErrorDetection method), 972	updatetext() (pynlpl.formats.folia.Suggestion method), 1005
updatetext() (pynlpl.formats.folia.Event method), 162 updatetext() (pynlpl.formats.folia.Example method), 175	updatetext() (pynlpl.formats.folia.SynsetFeature method), 882
updatetext() (pynlpl.formats.folia.Feature method), 870 updatetext() (pynlpl.formats.folia.Figure method), 188	updatetext() (pynlpl.formats.folia.SyntacticUnit method),
updatetext() (pynlpl.formats.folia.Gap method), 199 updatetext() (pynlpl.formats.folia.Head method), 212	updatetext() (pynlpl.formats.folia.SyntaxLayer method),
updatetext() (pynlpl.formats.folia.Headspan method), 796	updatetext() (pynlpl.formats.folia.Table method), 356
updatetext() (pynlpl.formats.folia.LangAnnotation method), 458	updatetext() (pynlpl.formats.folia.TableHead method), 382
updatetext() (pynlpl.formats.folia.LemmaAnnotation method), 469	updatetext() (pynlpl.formats.folia.Term method), 369 updatetext() (pynlpl.formats.folia.Text method), 396
updatetext() (pynlpl.formats.folia.Linebreak method), 225	updatetext() (pynlpl.formats.folia.TextContent method), 502
updatetext() (pynlpl.formats.folia.List method), 238	$update text()\ (pynlpl.formats.folia. Text Markup Correction$
updatetext() (pynlpl.formats.folia.ListItem method), 251	method), 848
updatetext() (pynlpl.formats.folia.Metric method), 1050 updatetext() (pynlpl.formats.folia.New method), 983	updatetext() (pynlpl.formats.folia.TextMarkupError method), 859
updatetext() (pynlpl.formats.folia.Note method), 264	updatetext() (pynlpl.formats.folia.TextMarkupGap
updatetext() (pynlpl.formats.folia.Observation method),	method), 816
573	updatetext() (pynlpl.formats.folia.TextMarkupString
updatetext() (pynlpl.formats.folia.ObservationLayer	method), 827
method), 702 updatetext() (pynlpl.formats.folia.Original method), 994	updatetext() (pynlpl.formats.folia.TextMarkupStyle method), 837
updatetext() (pynlpl.formats.folia.Paragraph method),	updatetext() (pynlpl.formats.folia.TimeSegment method),
277	643
updatetext() (pynlpl.formats.folia.Part method), 290 updatetext() (pynlpl.formats.folia.PhonContent method),	updatetext() (pynlpl.formats.folia.TimingLayer method), 761
513	updatetext() (pynlpl.formats.folia.Whitespace method),
updatetext() (pynlpl.formats.folia.PosAnnotation	409
method), 447 updatetext() (pynlpl.formats.folia.Predicate method), 584	updatetext() (pynlpl.formats.folia.Word method), 424
updatetext() (pynlpl.formats.folia.Quote method), 303	V
updatetext() (pynlpl.formats.folia.Reference method), 316	validate() (pynlpl.formats.sonar.CorpusDocumentX method), 1054
updatetext() (pynlpl.formats.folia.Row method), 329	values() (pynlpl.statistics.Distribution method), 1062
updatetext() (pynlpl.formats.folia.SemanticRole method),	values() (pynlpl.statistics.FrequencyList method), 1063
631	vector_add() (in module pynlpl.statistics), 1064
updatetext() (pynlpl.formats.folia.SemanticRolesLayer method), 749	visited() (pynlpl.search.AbstractSearch method), 1059
updatetext() (pynlpl.formats.folia.SenseAnnotation	viterbi() (pynlpl.statistics.HiddenMarkovModel method), 1063
method), 480	
updatetext() (pynlpl.formats.folia.Sentence method), 343	W
updatetext() (pynlpl.formats.folia.Sentiment method), 596	wait() (pynlpl.evaluation.AbstractExperiment method), 9
updatetext() (pynlpl.formats.folia.SentimentLayer	walk() (pynlpl.datatypes.Trie method), 7
method), 714	Whitespace (class in pynlpl.formats.folia), 396 Windower (class in pynlpl.textprocessors), 1066
updatetext() (pynlpl.formats.folia.Statement method),	Word (class in pynlpl.formats.folia), 409

WordAlignment (class in pynlpl.formats.giza), 1054	X
words() (pynlpl.formats.folia.AbstractStructureElement method), 47	XLINK (pynlpl.formats.folia.AbstractAnnotationLayer attribute), 75
words() (pynlpl.formats.folia.Cell method), 110 words() (pynlpl.formats.folia.Definition method), 123	XLINK (pynlpl.formats.folia.AbstractElement attribute), 26
words() (pynlpl.formats.folia.Division method), 136 words() (pynlpl.formats.folia.Document method), 20	XLINK (pynlpl.formats.folia.AbstractSpanAnnotation attribute), 52
words() (pynlpl.formats.folia.Entry method), 149 words() (pynlpl.formats.folia.Event method), 162	XLINK (pynlpl.formats.folia.AbstractStructureElement attribute), 37
words() (pynlpl.formats.folia.Example method), 175 words() (pynlpl.formats.folia.Figure method), 188	XLINK (pynlpl.formats.folia.AbstractTextMarkup attribute), 86
words() (pynlpl.formats.folia.Head method), 212 words() (pynlpl.formats.folia.Linebreak method), 225	XLINK (pynlpl.formats.folia.AbstractTokenAnnotation attribute), 63
words() (pynlpl.formats.folia.List method), 238 words() (pynlpl.formats.folia.ListItem method), 251	XLINK (pynlpl.formats.folia.ActorFeature attribute), 885 XLINK (pynlpl.formats.folia.Alignment attribute), 1009
words() (pynlpl.formats.folia.Note method), 264 words() (pynlpl.formats.folia.Paragraph method), 277	XLINK (pynlpl.formats.folia.AlignReference attribute),
words() (pynlpl.formats.folia.Part method), 290 words() (pynlpl.formats.folia.Quote method), 303	XLINK (pynlpl.formats.folia.Alternative attribute), 919 XLINK (pynlpl.formats.folia.AlternativeLayers at-
words() (pynlpl.formats.folia.Reference method), 316 words() (pynlpl.formats.folia.Row method), 329	tribute), 931  XLINK (pynlpl.formats.folia.BegindatetimeFeature at-
words() (pynlpl.formats.folia.Sentence method), 344 words() (pynlpl.formats.folia.Table method), 357	tribute), 896  XLINK (pynlpl.formats.folia.Cell attribute), 101
words() (pynlpl.formats.folia.TableHead method), 383 words() (pynlpl.formats.folia.Term method), 370	XLINK (pynlpl.formats.folia.Chunk attribute), 518 XLINK (pynlpl.formats.folia.ChunkingLayer attribute),
words() (pynlpl.formats.folia.Text method), 396 words() (pynlpl.formats.folia.Whitespace method), 409	647
words() (pynlpl.formats.folia.Word method), 424	XLINK (pynlpl.formats.folia.CoreferenceChain attribute), 530
words() (pynlpl.formats.sonar.CorpusDocument method), 1054	XLINK (pynlpl.formats.folia.CoreferenceLayer attribute), 658
words() (pynlpl.formats.sonar.CorpusDocumentX method), 1055	XLINK (pynlpl.formats.folia.CoreferenceLink attribute), 764
WPSParamSearch (class in pynlpl.evaluation), 10 wrefs() (pynlpl.formats.folia.AbstractSpanAnnotation	XLINK (pynlpl.formats.folia.Correction attribute), 944
wrefs() (pynlpl.formats.folia.AbstractSpanAnnotation method), 60	XLINK (pynlpl formats folia Definition ettribute), 954
wrefs() (pynlpl.formats.folia.Chunk method), 526	XLINK (pynlpl.formats.folia.Definition attribute), 114 XLINK (pynlpl.formats.folia.DependenciesLayer at-
wrefs() (pynlpl.formats.folia.CoreferenceChain method),	tribute), 670
538	XLINK (pynlpl.formats.folia.Dependency attribute), 541
wrefs() (pynlpl.formats.folia.CoreferenceLink method), 773	XLINK (pynlpl.formats.folia.DependencyDependent attribute), 776
wrefs() (pynlpl.formats.folia.Dependency method), 550 wrefs() (pynlpl.formats.folia.DependencyDependent	XLINK (pynlpl.formats.folia.Description attribute), 1032
method), 785	XLINK (pynlpl.formats.folia.Division attribute), 126 XLINK (pynlpl.formats.folia.DomainAnnotation at-
wrefs() (pynlpl.formats.folia.Entity method), 561	tribute), 429
wrefs() (pynlpl.formats.folia.Headspan method), 796 wrefs() (pynlpl.formats.folia.Observation method), 573	XLINK (pynlpl.formats.folia.EnddatetimeFeature at-
wrefs() (pynlpl.formats.folia.Predicate method), 585	tribute), 907
wrefs() (pynlpl.formats.folia.SemanticRole method), 631	XLINK (pynlpl.formats.folia.EntitiesLayer attribute), 682
wrefs() (pynlpl.formats.folia.Sentiment method), 596	XLINK (pynlpl.formats.folia.Entity attribute), 553
wrefs() (pynlpl.formats.folia.Statement method), 608	XLINK (pynlpl.formats.folia.Entry attribute), 139
wrefs() (pynlpl.formats.folia.SyntacticUnit method), 620	XLINK (pynlpl.formats.folia.ErrorDetection attribute),
wrefs() (pynlpl.formats.folia.TimeSegment method), 643	965
write() (pynlpl.formats.taggerdata.Taggerdata method), 1055	XLINK (pynlpl.formats.folia.Event attribute), 152 XLINK (pynlpl.formats.folia.Example attribute), 165

- XLINK (pynlpl.formats.folia.Feature attribute), 863 XLINK (pynlpl.formats.folia.Figure attribute), 178 XLINK (pynlpl.formats.folia.Gap attribute), 191 XLINK (pynlpl.formats.folia.Head attribute), 203 XLINK (pynlpl.formats.folia.Headspan attribute), 788 XLINK (pynlpl.formats.folia.LangAnnotation attribute), XLINK (pynlpl.formats.folia.LemmaAnnotation attribute), 462 XLINK (pynlpl.formats.folia.Linebreak attribute), 216 XLINK (pynlpl.formats.folia.List attribute), 228 XLINK (pynlpl.formats.folia.ListItem attribute), 241 XLINK (pynlpl.formats.folia.Metric attribute), 1043 XLINK (pynlpl.formats.folia.New attribute), 976 XLINK (pynlpl.formats.folia.Note attribute), 254 XLINK (pynlpl.formats.folia.Observation attribute), 565 XLINK (pynlpl.formats.folia.ObservationLayer tribute), 694 XLINK (pynlpl.formats.folia.Original attribute), 987 XLINK (pynlpl.formats.folia.Paragraph attribute), 267 XLINK (pynlpl.formats.folia.Part attribute), 280 XLINK (pynlpl.formats.folia.PhonContent attribute), 506 XLINK (pynlpl.formats.folia.PosAnnotation attribute), 440 XLINK (pynlpl.formats.folia.Predicate attribute), 576 XLINK (pynlpl.formats.folia.Ouote attribute), 293 XLINK (pynlpl.formats.folia.Reference attribute), 306 XLINK (pynlpl.formats.folia.Row attribute), 319 XLINK (pynlpl.formats.folia.SemanticRole attribute), 623 XLINK (pynlpl.formats.folia.SemanticRolesLayer attribute), 741 XLINK (pynlpl.formats.folia.SenseAnnotation attribute), XLINK (pynlpl.formats.folia.Sentence attribute), 333 XLINK (pynlpl.formats.folia.Sentiment attribute), 588 XLINK (pynlpl.formats.folia.SentimentLayer attribute), XLINK (pynlpl.formats.folia.Statement attribute), 600 XLINK (pynlpl.formats.folia.StatementLayer attribute), XLINK (pynlpl.formats.folia.SubjectivityAnnotation attribute), 484 XLINK (pynlpl.formats.folia.Suggestion attribute), 998 XLINK (pynlpl.formats.folia.SynsetFeature attribute), 874 XLINK (pynlpl.formats.folia.SyntacticUnit attribute), XLINK (pynlpl.formats.folia.SyntaxLayer attribute), 729 XLINK (pynlpl.formats.folia.Table attribute), 347 XLINK (pynlpl.formats.folia.TableHead attribute), 373 XLINK (pynlpl.formats.folia.Term attribute), 360 XLINK (pynlpl.formats.folia.Text attribute), 386 XLINK (pynlpl.formats.folia.TextContent attribute), 495
- XLINK (pynlpl.formats.folia.TextMarkupCorrection attribute), 841 XLINK (pynlpl.formats.folia.TextMarkupError attribute), XLINK (pynlpl.formats.folia.TextMarkupGap attribute), XLINK (pynlpl.formats.folia.TextMarkupString tribute), 819 XLINK (pynlpl.formats.folia.TextMarkupStyle attribute), 830 XLINK (pynlpl.formats.folia.TimeSegment attribute), XLINK (pynlpl.formats.folia.TimingLayer attribute), 753 XLINK (pynlpl.formats.folia.Whitespace attribute), 399 XLINK (pynlpl.formats.folia.Word attribute), 413 (pynlpl.formats.folia.AbstractAnnotationLayer xml() method), 83 xml() (pynlpl.formats.folia.AbstractElement method), 34 (pynlpl.formats.folia.AbstractSpanAnnotation xml() method), 60 xml() (pynlpl.formats.folia.AbstractStructureElement method), 47 (pynlpl.formats.folia.AbstractTextMarkup xml() method), 94 (pynlpl.formats.folia.AbstractTokenAnnotation xml() method), 71 xml() (pynlpl.formats.folia.ActorFeature method), 893 xml() (pynlpl.formats.folia.Alignment method), 1017 (pynlpl.formats.folia.AlignReference method), 1028 xml() (pynlpl.formats.folia.Alternative method), 928 xml() (pynlpl.formats.folia.AlternativeLayers method), (pynlpl.formats.folia.BegindatetimeFeature xml() method), 904 xml() (pynlpl.formats.folia.Cell method), 110 xml() (pynlpl.formats.folia.Chunk method), 526 xml() (pynlpl.formats.folia.ChunkingLayer method), 655 xml() (pynlpl.formats.folia.CoreferenceChain method), xml() (pynlpl.formats.folia.CoreferenceLayer method), (pynlpl.formats.folia.CoreferenceLink method), xml() xml() (pynlpl.formats.folia.Correction method), 950 xml() (pynlpl.formats.folia.Current method), 961 xml() (pynlpl.formats.folia.Definition method), 123 xml() (pynlpl.formats.folia.DependenciesLayer method),

xml() (pynlpl.formats.folia.Dependency method), 550

xml() (pynlpl.formats.folia.Description method), 1039

xml() (pynlpl.formats.folia.Division method), 136

method), 785

(pynlpl.formats.folia.DependencyDependent

Index 1183

xml()

xml() (pynlpl.formats.folia.Document method), 21	xml() (pynlpl.formats.folia.TableHead method), 383
xml() (pynlpl.formats.folia.DomainAnnotation method),	xml() (pynlpl.formats.folia.Term method), 370
437	xml() (pynlpl.formats.folia.Text method), 396
xml() (pynlpl.formats.folia.EnddatetimeFeature method),	xml() (pynlpl.formats.folia.TextContent method), 502
915	xml() (pynlpl.formats.folia.TextMarkupCorrection
xml() (pynlpl.formats.folia.EntitiesLayer method), 691	method), 848
xml() (pynlpl.formats.folia.Entity method), 561	xml() (pynlpl.formats.folia.TextMarkupError method),
xml() (pynlpl.formats.folia.Entry method), 149	859
xml() (pynlpl.formats.folia.ErrorDetection method), 973	xml() (pynlpl.formats.folia.TextMarkupGap method), 816
xml() (pynlpl.formats.folia.Event method), 162	xml() (pynlpl.formats.folia.TextMarkupString method),
xml() (pynlpl.formats.folia.Example method), 175	827
xml() (pynlpl.formats.folia.Feature method), 871	xml() (pynlpl.formats.folia.TextMarkupStyle method),
xml() (pynlpl.formats.folia.Figure method), 188	837
xml() (pynlpl.formats.folia.Gap method), 199	xml() (pynlpl.formats.folia.TimeSegment method), 643
xml() (pynlpl.formats.folia.Head method), 212	xml() (pynlpl.formats.folia.TimingLayer method), 761
xml() (pynlpl.formats.folia.Headspan method), 796	xml() (pynlpl.formats.folia.Whitespace method), 409
xml() (pynlpl.formats.folia.LangAnnotation method), 459	xml() (pynlpl.formats.folia.Word method), 424
xml() (pynlpl.formats.folia.LemmaAnnotation method),	xmldeclarations() (pynlpl.formats.folia.Document
470	method), 21
xml() (pynlpl.formats.folia.Linebreak method), 225	xmlmetadata() (pynlpl.formats.folia.Document method),
xml() (pynlpl.formats.folia.List method), 238	21
xml() (pynlpl.formats.folia.ListItem method), 251	xmlstring() (pynlpl.formats.folia.AbstractAnnotationLayer
xml() (pynlpl.formats.folia.Metric method), 1051	method), 83
xml() (pynlpl.formats.folia.New method), 984	xmlstring() (pynlpl.formats.folia.AbstractElement
xml() (pynlpl.formats.folia.Note method), 264	method), 34
xml() (pynlpl.formats.folia.Observation method), 573	xmlstring() (pynlpl.formats.folia.AbstractSpanAnnotation
xml() (pynlpl.formats.folia.ObservationLayer method),	method), 60
702	xmlstring() (pynlpl.formats.folia.AbstractStructureElement
xml() (pynlpl.formats.folia.Original method), 995	method), 47
xml() (pynlpl.formats.folia.Paragraph method), 277	xmlstring() (pynlpl.formats.folia.AbstractTextMarkup
xml() (pynlpl.formats.folia.Part method), 290	method), 94
xml() (pynlpl.formats.folia.PhonContent method), 513	xmlstring() (pynlpl.formats.folia.AbstractTokenAnnotation
xml() (pynlpl.formats.folia.PosAnnotation method), 448	method), 71
xml() (pynlpl.formats.folia.Predicate method), 585	xmlstring() (pynlpl.formats.folia.ActorFeature method),
xml() (pynlpl.formats.folia.Quote method), 303	893
xml() (pynlpl.formats.folia.Reference method), 316	xmlstring() (pynlpl.formats.folia.Alignment method),
xml() (pynlpl.formats.folia.Row method), 329	Amistring() (pympinormatshonan mgmment method);
	1017
	1017
xml() (pynlpl.formats.folia.SemanticRole method), 631	1017 xmlstring() (pynlpl.formats.folia.AlignReference
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer	1017 xmlstring() (pynlpl.formats.folia.AlignReference method), 1028
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749	1017 xmlstring() (pynlpl.formats.folia.AlignReference
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer	1017  xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608 xml() (pynlpl.formats.folia.StatementLayer method), 726	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110  xmlstring() (pynlpl.formats.folia.Chunk method), 526
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.StatementLayer method), 726	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110  xmlstring() (pynlpl.formats.folia.Chunk method), 526  xmlstring() (pynlpl.formats.folia.ChunkingLayer
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.SubjectivityAnnotation method), 492	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110  xmlstring() (pynlpl.formats.folia.Chunk method), 526  xmlstring() (pynlpl.formats.folia.ChunkingLayer method), 655
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.SubjectivityAnnotation method), 492 xml() (pynlpl.formats.folia.Suggestion method), 1006	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110  xmlstring() (pynlpl.formats.folia.Chunk method), 526  xmlstring() (pynlpl.formats.folia.ChunkingLayer method), 655  xmlstring() (pynlpl.formats.folia.CoreferenceChain
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.SubjectivityAnnotation method), 492 xml() (pynlpl.formats.folia.Suggestion method), 1006 xml() (pynlpl.formats.folia.SynsetFeature method), 882	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110  xmlstring() (pynlpl.formats.folia.Chunk method), 526  xmlstring() (pynlpl.formats.folia.ChunkingLayer method), 655  xmlstring() (pynlpl.formats.folia.CoreferenceChain method), 538
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.SubjectivityAnnotation method), 492 xml() (pynlpl.formats.folia.Suggestion method), 1006 xml() (pynlpl.formats.folia.SynsetFeature method), 882 xml() (pynlpl.formats.folia.SyntacticUnit method), 620	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110  xmlstring() (pynlpl.formats.folia.Chunk method), 526  xmlstring() (pynlpl.formats.folia.ChunkingLayer method), 655  xmlstring() (pynlpl.formats.folia.CoreferenceChain method), 538  xmlstring() (pynlpl.formats.folia.CoreferenceLayer
xml() (pynlpl.formats.folia.SemanticRole method), 631 xml() (pynlpl.formats.folia.SemanticRolesLayer method), 749 xml() (pynlpl.formats.folia.SenseAnnotation method), 481 xml() (pynlpl.formats.folia.Sentence method), 344 xml() (pynlpl.formats.folia.Sentiment method), 596 xml() (pynlpl.formats.folia.SentimentLayer method), 714 xml() (pynlpl.formats.folia.Statement method), 608 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.StatementLayer method), 726 xml() (pynlpl.formats.folia.SubjectivityAnnotation method), 492 xml() (pynlpl.formats.folia.Suggestion method), 1006 xml() (pynlpl.formats.folia.SynsetFeature method), 882	xmlstring() (pynlpl.formats.folia.AlignReference method), 1028  xmlstring() (pynlpl.formats.folia.Alternative method), 928  xmlstring() (pynlpl.formats.folia.AlternativeLayers method), 939  xmlstring() (pynlpl.formats.folia.BegindatetimeFeature method), 904  xmlstring() (pynlpl.formats.folia.Cell method), 110  xmlstring() (pynlpl.formats.folia.Chunk method), 526  xmlstring() (pynlpl.formats.folia.ChunkingLayer method), 655  xmlstring() (pynlpl.formats.folia.CoreferenceChain method), 538

- method), 773 xmlstring() (pynlpl.formats.folia.Correction method), 951 xmlstring() (pynlpl.formats.folia.Current method), 962 xmlstring() (pynlpl.formats.folia.Definition method), 123 (pynlpl.formats.folia.DependenciesLayer xmlstring() method), 679 xmlstring() (pynlpl.formats.folia.Dependency method), xmlstring() (pynlpl.formats.folia.DependencyDependent method), 785 xmlstring() (pynlpl.formats.folia.Description method), xmlstring() (pynlpl.formats.folia.Division method), 136 xmlstring() (pynlpl.formats.folia.Document method), 21 (pynlpl.formats.folia.DomainAnnotation xmlstring() method), 437 xmlstring() (pynlpl.formats.folia.EnddatetimeFeature method), 915 xmlstring() (pynlpl.formats.folia.EntitiesLayer method), xmlstring() (pynlpl.formats.folia.Entity method), 561 xmlstring() (pynlpl.formats.folia.Entry method), 149 xmlstring() (pynlpl.formats.folia.ErrorDetection method), xmlstring() (pynlpl.formats.folia.Event method), 162 xmlstring() (pynlpl.formats.folia.Example method), 175 xmlstring() (pynlpl.formats.folia.Feature method), 871 xmlstring() (pynlpl.formats.folia.Figure method), 188 xmlstring() (pynlpl.formats.folia.Gap method), 199 xmlstring() (pynlpl.formats.folia.Head method), 212 xmlstring() (pynlpl.formats.folia.Headspan method), 796 xmlstring() (pynlpl.formats.folia.LangAnnotation method), 459 (pynlpl.formats.folia.LemmaAnnotation xmlstring() method), 470 xmlstring() (pynlpl.formats.folia.Linebreak method), 225 xmlstring() (pynlpl.formats.folia.List method), 238 xmlstring() (pynlpl.formats.folia.ListItem method), 251 xmlstring() (pynlpl.formats.folia.Metric method), 1051 xmlstring() (pynlpl.formats.folia.New method), 984 xmlstring() (pynlpl.formats.folia.Note method), 264 xmlstring() (pynlpl.formats.folia.Observation method), xmlstring() (pynlpl.formats.folia.ObservationLayer method), 702 xmlstring() (pynlpl.formats.folia.Original method), 995 xmlstring() (pynlpl.formats.folia.Paragraph method), 277 xmlstring() (pynlpl.formats.folia.Part method), 290 xmlstring() (pynlpl.formats.folia.PhonContent method), 513 xmlstring() (pynlpl.formats.folia.PosAnnotation method), xmlstring() (pynlpl.formats.folia.Predicate method), 585 xmlstring() (pynlpl.formats.folia.Quote method), 303
- xmlstring() (pynlpl.formats.folia.Reference method), 316 xmlstring() (pynlpl.formats.folia.Row method), 329 xmlstring() (pynlpl.formats.folia.SemanticRole method), 631
- xmlstring() (pynlpl.formats.folia.SemanticRolesLayer method), 749
- xmlstring() (pynlpl.formats.folia.SenseAnnotation method), 481
- xmlstring() (pynlpl.formats.folia.Sentence method), 344 xmlstring() (pynlpl.formats.folia.Sentiment method), 596 xmlstring() (pynlpl.formats.folia.SentimentLayer method), 714
- xmlstring() (pynlpl.formats.folia.Statement method), 608 xmlstring() (pynlpl.formats.folia.StatementLayer method), 726
- xmlstring() (pynlpl.formats.folia.SubjectivityAnnotation method), 492
- $xmlstring() \quad (pynlpl.formats.folia.Suggestion \quad method), \\ 1006$
- xmlstring() (pynlpl.formats.folia.SynsetFeature method), 882
- xmlstring() (pynlpl.formats.folia.SyntacticUnit method), 620
- xmlstring() (pynlpl.formats.folia.SyntaxLayer method), 738
- xmlstring() (pynlpl.formats.folia.Table method), 357 xmlstring() (pynlpl.formats.folia.TableHead method), 383
- xmlstring() (pynlpl.formats.folia.Term method), 370 xmlstring() (pynlpl.formats.folia.Text method), 396 xmlstring() (pynlpl.formats.folia.TextContent method),
- xmlstring() (pynlpl.formats.folia.TextMarkupCorrection method), 848
- xmlstring() (pynlpl.formats.folia.TextMarkupError method), 859
- xmlstring() (pynlpl.formats.folia.TextMarkupGap method), 816
- xmlstring() (pynlpl.formats.folia.TextMarkupString method), 827
- xmlstring() (pynlpl.formats.folia.TextMarkupStyle method), 837
- xmlstring() (pynlpl.formats.folia.TimeSegment method), 643
- xmlstring() (pynlpl.formats.folia.TimingLayer method), 761
- xmlstring() (pynlpl.formats.folia.Whitespace method), 409
- xmlstring() (pynlpl.formats.folia.Word method), 424 XMLTAG (pynlpl.formats.folia.AbstractAnnotationLayer attribute), 75
- XMLTAG (pynlpl.formats.folia.AbstractElement attribute), 26
- $XMLTAG\ (pynlpl.formats.folia. Abstract Span Annotation$

attribute), 52 XMLTAG (pynlpl.formats.folia.AbstractStructureElement XMLTAG (pynlpl.formats.folia.Gap attribute), 191 attribute), 37 XMLTAG (pynlpl.formats.folia.AbstractTextMarkup attribute), 86 XMLTAG (pynlpl.formats.folia.AbstractTokenAnnotation attribute), 63 XMLTAG (pynlpl.formats.folia.ActorFeature attribute), XMLTAG (pynlpl.formats.folia.Alignment attribute), (pynlpl.formats.folia.AlignReference XMLTAG tribute), 1020 XMLTAG (pynlpl.formats.folia.Alternative attribute), 919 XMLTAG (pynlpl.formats.folia.AlternativeLayers tribute), 931 XMLTAG (pynlpl.formats.folia.BegindatetimeFeature attribute), 896 XMLTAG (pynlpl.formats.folia.Cell attribute), 101 XMLTAG (pynlpl.formats.folia.Chunk attribute), 518 (pynlpl.formats.folia.ChunkingLayer tribute), 647 **XMLTAG** (pynlpl.formats.folia.CoreferenceChain attribute), 530 XMLTAG (pynlpl.formats.folia.CoreferenceLayer tribute), 658 (pynlpl.formats.folia.CoreferenceLink XMLTAG tribute), 764 XMLTAG (pynlpl.formats.folia.Correction attribute), 944 XMLTAG (pynlpl.formats.folia.Current attribute), 954 XMLTAG (pynlpl.formats.folia.Definition attribute), 114 XMLTAG (pynlpl.formats.folia.DependenciesLayer attribute), 670 XMLTAG (pynlpl.formats.folia.Dependency attribute), XMLTAG (pynlpl.formats.folia.DependencyDependent attribute), 776 XMLTAG (pynlpl.formats.folia.Description attribute), 1032 XMLTAG (pynlpl.formats.folia.Division attribute), 127 XMLTAG (pynlpl.formats.folia.DomainAnnotation attribute), 429 XMLTAG (pynlpl.formats.folia.EnddatetimeFeature attribute), 907 XMLTAG (pynlpl.formats.folia.EntitiesLayer attribute), XMLTAG (pynlpl.formats.folia.Entity attribute), 553 XMLTAG (pynlpl.formats.folia.Entry attribute), 139 XMLTAG (pynlpl.formats.folia.ErrorDetection attribute),

965

XMLTAG (pynlpl.formats.folia.Event attribute), 153 XMLTAG (pynlpl.formats.folia.Example attribute), 165

XMLTAG (pynlpl.formats.folia.Feature attribute), 863

XMLTAG (pynlpl.formats.folia.Figure attribute), 179 XMLTAG (pynlpl.formats.folia.Head attribute), 203 XMLTAG (pynlpl.formats.folia.Headspan attribute), 788 (pynlpl.formats.folia.LangAnnotation XMLTAG tribute), 451 XMLTAG (pynlpl.formats.folia.LemmaAnnotation attribute), 462 XMLTAG (pynlpl.formats.folia.Linebreak attribute), 216 XMLTAG (pynlpl.formats.folia.List attribute), 228 XMLTAG (pynlpl.formats.folia.ListItem attribute), 241 XMLTAG (pynlpl.formats.folia.Metric attribute), 1043 XMLTAG (pynlpl.formats.folia.New attribute), 976 XMLTAG (pynlpl.formats.folia.Note attribute), 254 XMLTAG (pynlpl.formats.folia.Observation attribute), 565 XMLTAG (pynlpl.formats.folia.ObservationLayer attribute), 694 XMLTAG (pynlpl.formats.folia.Original attribute), 987 XMLTAG (pynlpl.formats.folia.Paragraph attribute), 268 XMLTAG (pynlpl.formats.folia.Part attribute), 281 XMLTAG (pynlpl.formats.folia.PhonContent attribute), XMLTAG (pynlpl.formats.folia.PosAnnotation attribute), 440 XMLTAG (pynlpl.formats.folia.Predicate attribute), 576 XMLTAG (pynlpl.formats.folia.Quote attribute), 293 XMLTAG (pynlpl.formats.folia.Reference attribute), 307 XMLTAG (pynlpl.formats.folia.Row attribute), 319 XMLTAG (pynlpl.formats.folia.SemanticRole attribute), XMLTAG (pynlpl.formats.folia.SemanticRolesLayer attribute), 741 XMLTAG (pynlpl.formats.folia.SenseAnnotation tribute), 473 XMLTAG (pynlpl.formats.folia.Sentence attribute), 333 XMLTAG (pynlpl.formats.folia.Sentiment attribute), 588 (pynlpl.formats.folia.SentimentLayer XMLTAG tribute), 706 XMLTAG (pynlpl.formats.folia.Statement attribute), 600 (pynlpl.formats.folia.StatementLaver tribute), 717 XMLTAG (pynlpl.formats.folia.SubjectivityAnnotation attribute), 484 XMLTAG (pynlpl.formats.folia.Suggestion attribute), XMLTAG (pynlpl.formats.folia.SynsetFeature attribute), XMLTAG (pynlpl.formats.folia.SyntacticUnit attribute), XMLTAG (pynlpl.formats.folia.SyntaxLayer attribute),

XMLTAG (pynlpl.formats.folia.Table attribute), 347

XMLTAG (pynlpl.formats.folia.TableHead attribute), 373

- XMLTAG (pynlpl.formats.folia.Term attribute), 360
- XMLTAG (pynlpl.formats.folia.Text attribute), 386
- XMLTAG (pynlpl.formats.folia.TextContent attribute), 495
- XMLTAG (pynlpl.formats.folia.TextMarkupCorrection attribute), 841
- XMLTAG (pynlpl.formats.folia.TextMarkupError attribute), 851
- XMLTAG (pynlpl.formats.folia.TextMarkupGap attribute), 809
- XMLTAG (pynlpl.formats.folia.TextMarkupString attribute), 819
- XMLTAG (pynlpl.formats.folia.TextMarkupStyle attribute), 830
- XMLTAG (pynlpl.formats.folia.TimeSegment attribute), 635
- XMLTAG (pynlpl.formats.folia.TimingLayer attribute), 753
- XMLTAG (pynlpl.formats.folia.Whitespace attribute), 399
- $XMLTAG\ (pynlpl.formats.folia.Word\ attribute),\ 413$
- xpath() (pynlpl.formats.folia.Document method), 21
- xpath() (pynlpl.formats.sonar.CorpusDocumentX method), 1055