Bacalhau Documentation

Release 1.0

Miguel Vieira

Sep 27, 2017

Contents

Bacalhau is a Python library and a tool to automatically generate topic hierarchies, from a corpus of texts, using WordNet.

CHAPTER 1

Requirements

- Natural Language Toolkit
- NetworkX
- lxml
- PyGraphviz

CHAPTER 2

Support

See the issue tracker.

CHAPTER 3

Contents

Installation

To install bacalhau you first need to download or clone it from the GitHub repository. To clone bacalhau, open a Terminal, go to a directory of your choice and run:

```
git clone https://github.com/kcl-ddh/bacalhau.git
```

To update a previous version, go to the directory where bacalhau is cloned, and run:

```
git pull
```

To install bacalhau into your system, first install the requirements:

```
pip install -r requirements.txt
```

After all the requiremnts are installed, install the NLTK data. Once all the requirements have been installed run the bacalhau setup script:

```
python setup.py install
```

To verify that bacalhau is installed, type bacalhau on a Terminal and you should see a message on how to use it. If you don't want to install bacalhau system wide, it can also be installed on a virtual environment.

Usage

To generate a topic hierarchy for a corpus, run the `bacalhau` script with the appropriate arguments. These are documented in the script; use `bacalhau -h` to see them.

Handling new document formats

If the corpus files are not TEI XML, an implementation of the `bacalhau.document.Document` class must be written. The name of this class (with complete package path; for example, `bacalhau.tei_document.TEIDocument`) is passed to the `bacalhau` script with `--document` option.

Corpora with documents of more than a single type are not supported.

Library Reference

Classes

bacalhau.corpus.Corpus

class `bacalhau.corpus.`**`Corpus`**(*corpus_path*, *document_class*, *tokenizer=<WordPunctTokenizer object>*, *stopwords=<Mock object>*, ***document_kwargs*)

Bases: `object`

A manager class to generate topic hierarchies from files.

Creates a new *`Corpus`* for the given path, using the given *`bacalhau.document.Document`* class to process the files.

Parameters

- **`corpus_path`** (`str`) – path to the files.
- **`document_class`** (*`bacalhau.document.Document`*) – document class used to process the corpus files.
- **`tokenizer`** (`nltk.tokenize.api.TokenizerI`) – tokenizer used to tokenize the files in the corpus, defaults to `nltk.tokenize.regexp.WordPunctTokenizer`.
- **`stopwords`** (`list`) – words to be removed from the texts, defaults to `nltk.corpus.stopwords.words('english')`.

`_add_tf_idf`(*term_data*)

Returns `term_data` with a TF.IDF value added to each term/text combination.

Parameters **`term_data`** (`dict`) – dict with term/text combination.

Return type `dict`

`_get_documents`()

Creates a *`bacalhau.document.Document`* object for each of the files in the corpus, and returns them in a `list`.

Parameters **`corpus_path`** (`str`) – path to the corpus files.

Returns documents in this corpus.

Return type `list`

`_get_hypernym`(*word*)

Returns a list of the hypernyms for the given word.

Parameters **`word`** (`str`) – the word to get the hypernym for.

Return type `list`

_get_term_data()
Returns term data for all of the *bacalhau.document.Document* objects in this corpus.

Return type dict

_get_text_count()
Returns the number of *bacalhau.text.Text* objects in this corpus.

Return type float

annotate_topic_tree(*tree*)
Annotates the nodes in the *bacalhau.topic_tree.TopicTree* with information about which *bacalhau.text.Text* and counts the nodes relate to.

Parameters **tree** (*bacalhau.topic_tree.TopicTree*) – topic tree of terms

Return type *bacalhau.topic_tree.TopicTree*

generate_topic_tree(*n_terms*)
Generates a *bacalhau.topic_tree.TopicTree* for the corpus, using a maximum of n_terms from each *bacalhau.text.Text*. First extracts top terms; second gets hypernyms for each of the terms; third creates the *bacalhau.topic_tree.TopicTree* using the hypernyms.

Parameters **n_terms** (int) – maximum number of terms to be used from each Text.

Returns the generated topic tree.

Return type *bacalhau.topic_tree.TopicTree*

get_hypernyms(*top_terms*)
Returns a dictionary with the hypernyms for the given terms.

Parameters **top_terms** (dict) – dict with term/text information.

Returns {text: {term: hypernym}}.

Return type dict

get_top_terms(*n_terms*)
Returns a dictionary with the highest n_terms for each *bacalhau.text.Text* from the term data dictionary.

Parameters **n_terms** (int) – maximum number of terms to be used from each text.

Returns dict

get_topic_tree(*hypernyms*)
Generates and returns a *bacalhau.topic_tree.TopicTree* for the given hypernyms.

Parameters **hypernyms** (dict) – dictionary of hypernyms.

Return type *bacalhau.topic_tree.TopicTree*

bacalhau.document.Document

class bacalhau.document.**Document**(*filepath*, *tokenizer*, *stopwords*)
Bases: object

Abstract class to read from/write to files. Different implementations should extend this class and override the abstract methods.

Creates a new Document for the given file path.

Parameters

- **filepath** (str) – path to the file.
- **tokenizer** (nltk.tokenize.api.TokenizerI) – tokenizer used to tokenize the files in the corpus.
- **stopwords** (list) – words to be removed from the texts.

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 29

_abc_registry = <_weakrefset.WeakSet object>

get_term_data()
Returns term data for each *bacalhau.text.Text* within this document.

Returns dict

get_text_count()
Returns the number of *bacalhau.text.Text* objects for this *Document*.

Returns number of *bacalhau.text.Text* objects.

Return type int

get_texts()
Returns a list of *bacalhau.text.Text* objects within this document.

Returns list of *bacalhau.text.Text* objects.

Return type list

bacalhau.tei_document.TEIDocument

class bacalhau.tei_document.**TEIDocument**(*filepath*, *tokenizer*, *stopwords*, *xpath*, *ns_map={'xml': 'http://www.w3.org/XML/1998/namespace', 'tei': 'http://www.tei-c.org/ns/1.0'}*)
Bases: *bacalhau.document.Document*

Implementation of the abstract *bacalhau.document.Document* class to work with TEI files.

Creates a new *TEIDocument* for the given file path.

Parameters

- **filepath** (str) – path to the file.
- **tokenizer** (nltk.tokenize.api.TokenizerI) – tokenizer used to tokenize the files in the corpus.
- **stopwords** (list) – words to be removed from the texts.
- **xpath** (str) – XPath where to get the *bacalhau.text.Text* from the TEI files.
- **ns_map** (dict) – namespaces used in the *TEIDocument*.

NS_MAP = {'xml': 'http://www.w3.org/XML/1998/namespace', 'tei': 'http://www.tei-c.org/ns/1.0'}

TEI = '{http://www.tei-c.org/ns/1.0}'

TEI_NAMESPACE = 'http://www.tei-c.org/ns/1.0'

XML = '{http://www.w3.org/XML/1998/namespace}'

XML_NAMESPACE = 'http://www.w3.org/XML/1998/namespace'

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 29

_abc_registry = <_weakrefset.WeakSet object>

get_term_data()
Returns term data for each *bacalhau.text.Text* within this document.

Return type dict

get_texts()
Returns a list of *bacalhau.text.Text* objects within this document.

Returns *bacalhau.text.Text* objects within this document.

Return type list

bacalhau.text.Text

class bacalhau.text.**Text**(*text_id*, *content*, *tokenizer*, *stopwords*)
Bases: object

Represents a text unit from a *bacalhau.document.Document*.

Creates a new *Text* object.

Parameters

- **text_id** (str) – id of the *Text*.
- **content** (str) – content of the *Text*.
- **tokenizer** (nltk.tokenize.api.TokenizerI) – tokenizer used to tokenize the files in the corpus.
- **stopwords** (list of words.) – words to be removed from the texts.

_is_valid_token(*token*)
Checks if the token is suitable for processing. A token is suitable if: it is not in the list of stopwords; it is composed of alphabetical character; and is a considered a noun by WordNet.

Parameters **token** (str) – the token to validate.

Returns True if token is valid.

Return type bool

get_term_data()
Returns term data for this text.

The term's data are the unnormalised and normalised frequency counts of the term in this text. The former uses the "count" key, the latter "frequency".

The data is structured as a nested dictionary (term -> text -> counts) for easy merging of the term data from multiple *Text*s.

Return type dict

bacalhau.topic_tree.TopicTree

class bacalhau.topic_tree.**TopicTree**(*data=None*, ***attr*)
Bases: networkx.classes.digraph.DiGraph

Represents a TopicTree. Extends networkx.DiGraph.

Creates a new *TopicTree*.

Parameters

- **data** (list, *TopicTree* or any networkx graph object.) – data to initialize the tree with. If no data is supplied an empty tree is created.
- **attr** (*key/value pairs.*) – keyword arguments to add to the tree.

_eliminate_child_with_parent_name(*node*)
Eliminate a child node whose name appears within the parent's name.

Parameters **node** (*str.*) – name of node to process.

_eliminate_parents(*node*, *min_children*)
Recursively eliminates a parent of the current node that has fewer than min_children children, unless the parent is the root.

Parameters

- **node** (*str.*) – name of node to process.
- **min_children** (*int.*) – minimum number of children that a parent should have

compress(*min_children=2*)
Compresses the tree based on the castanet algorithm: 1. starting from the leaves, recursively eliminate a parent that has fewer than min_children, unless the parent is the root; 2. eliminate a child whose name appears within the parent's name.

Parameters **min_children** (*int.*) – minimum number of children that a parent should have, defaults to 2.

prune(*nodes*)
Removes the given nodes from the tree.

Parameters **nodes** (*list of str.*) – names of the nodes to be remove from the tree.

render(*filepath*, *format='svg'*, *prog='dot'*, *attributes={}*)
Renders the tree into the file at filepath.

filepath may also be a File-like object.

to_json(*filepath*)
Serializes the TopicTree to JSON Graph format and writes it to a file.

filepath is a file path or File-like object.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

Symbols

A

C

D

G

N

P

R

T

X