ternip Documentation

Release 1.1dev

Author

Mar 17, 2017

Contents

1 ternip Package **3**
- 1.1 ternip Package . . . 3
- 1.2 timex Module . . . 3
- 1.3 Subpackages . . . 3
  - 1.3.1 formats Package . . . 3
    - 1.3.1.1 formats Package . . . 3
    - 1.3.1.2 gate Module . . . 3
    - 1.3.1.3 tempeval2 Module . . . 4
    - 1.3.1.4 tern Module . . . 4
    - 1.3.1.5 timeml Module . . . 5
    - 1.3.1.6 timex2 Module . . . 5
    - 1.3.1.7 timex3 Module . . . 5
    - 1.3.1.8 xml_doc Module . . . 6
  - 1.3.2 rule_engine Package . . . 7
    - 1.3.2.1 expressions Module . . . 7
    - 1.3.2.2 normalisation_rule Module . . . 7
    - 1.3.2.3 normalisation_rule_block Module . . . 8
    - 1.3.2.4 normalisation_rule_engine Module . . . 8
    - 1.3.2.5 recognition_rule Module . . . 8
    - 1.3.2.6 recognition_rule_block Module . . . 9
    - 1.3.2.7 recognition_rule_engine Module . . . 9
    - 1.3.2.8 rule Module . . . 9
    - 1.3.2.9 rule_block Module . . . 9
    - 1.3.2.10 rule_engine Module . . . 9
    - 1.3.2.11 Subpackages . . . 10

2 Indices and tables **13**

Python Module Index **15**

Contents:

CHAPTER 1

ternip Package

ternip Package

ternip.__init__.**normaliser**()
Returns default normaliser, already configured.

ternip.__init__.**recogniser**()
Returns the default recogniser, already configured.

timex Module

class ternip.timex.**Timex**(*type=None*, *value=None*, *id=None*)
Bases: object

A temporal expression

ternip.timex.**add_timex_ids**(*ts*)
Goes through all TIMEXs and adds IDs to the timexes, if they don't exist already. Each ID is an integer, and is guaranteed to be unique in this set of timexes.

Subpackages

formats Package

formats Package

gate Module

class ternip.formats.gate.**GateDocument**(*file*)
Bases: object

A class to facilitate communication with GATE

get_dct_sents()
Returns the creation time sents for this document.

get_sents()
Returns a representation of this document in the [[(word, pos, timexes), ...], ...] format.

reconcile(*sents*)
Update this document with the newly annotated tokens.

reconcile_dct(*dct*)
Adds a TIMEX to the DCT tag and return the DCT

tempeval2 Module

class ternip.formats.tempeval2.**TempEval2Document**(*file*, *docid=''*, *dct='XXXXXXXX'*)
Bases: object

A class which uses the format of stand-off format of TempEval-2

static create(*sents*, *docid=''*)
Creates a TempEval-2 document from the internal representation

sents is the [[(word, pos, timexes), ...], ...] format.

get_attrs()
Print out the format suitable for timex-attributes.tab

get_dct_sents()
Returns the creation time sents for this document.

get_extents()
Print out the format suitable for timex-extents.tab

get_sents()
Returns a representation of this document in the [[(word, pos, timexes), ...], ...] format.

static load_multi(*file*, *dct_file*)
Load multiple documents from a single base-segmentation.tab

reconcile(*sents*)
Update this document with the newly annotated tokens.

reconcile_dct(*dct*)
Adds a TIMEX to the DCT tag and return the DCT

tern Module

class ternip.formats.tern.**TernDocument**(*file*, *nodename='TEXT'*, *has_S=False*, *has_LEX=False*, *pos_attr=False*)
Bases: *ternip.formats.timex2.Timex2XmlDocument*

A class which can handle TERN documents

static create(*sents*, *docid*, *tok_offsets=None*, *add_S=False*, *add_LEX=False*, *pos_attr=False*, *dct=''*)
Creates a TERN document from the internal representation

sents is the [[(word, pos, timexes), ...], ...] format.

tok_offsets is used to correctly reinsert whitespace lost in tokenisation. It's in the format of a list of lists of integers, where each integer is the offset from the start of the sentence of that token. If set to None (the default), then a single space is assumed between all tokens.

If add_S is set to something other than false, then the tags to indicate sentence boundaries are added, with the name of the tag being the value of add_S

add_LEX is similar, but for token boundaries

pos_attr is similar but refers to the name of the attribute on the LEX (or whatever) tag that holds the POS tag.

dct is the document creation time string

get_dct_sents()
Returns the creation time sents for this document.

reconcile_dct(*dct*, *add_S=False*, *add_LEX=False*, *pos_attr=False*)
Adds a TIMEX to the DCT tag and return the DCT

timeml Module

class ternip.formats.timeml.**TimeMlDocument**(*file*, *nodename=None*, *has_S=False*, *has_LEX=False*, *pos_attr=False*)
Bases: *ternip.formats.timex3.Timex3XmlDocument*

A class which holds a TimeML representation of a document.

Suitable for use with the AQUAINT dataset.

static create(*sents*, *tok_offsets=None*, *add_S=False*, *add_LEX=False*, *pos_attr=False*)
Creates a TimeML document from the internal representation

sents is the [[(word, pos, timexes), ...], ...] format.

tok_offsets is used to correctly reinsert whitespace lost in tokenisation. It's in the format of a list of lists of integers, where each integer is the offset from the start of the sentence of that token. If set to None (the default), then a single space is assumed between all tokens.

If add_S is set to something other than false, then the tags to indicate sentence boundaries are added, with the name of the tag being the value of add_S

add_LEX is similar, but for token boundaries

pos_attr is similar but refers to the name of the attribute on the LEX (or whatever) tag that holds the POS tag.

timex2 Module

class ternip.formats.timex2.**Timex2XmlDocument**(*file*, *nodename=None*, *has_S=False*, *has_LEX=False*, *pos_attr=False*)
Bases: *ternip.formats.xml_doc.XmlDocument*

A class which takes any random XML document and adds TIMEX2 tags to it.

timex3 Module

class ternip.formats.timex3.**Timex3XmlDocument**(*file*, *nodename=None*, *has_S=False*, *has_LEX=False*, *pos_attr=False*)
Bases: *ternip.formats.xml_doc.XmlDocument*

A class which takes any random XML document and adds TIMEX3 tags to it.

Suitable for use with Timebank, which contains many superfluous tags that aren't in the TimeML spec, even though it claims to be TimeML.

xml_doc Module

exception ternip.formats.xml_doc.**BadNodeNameError**
Bases: exceptions.Exception

exception ternip.formats.xml_doc.**NestingError**(*s*)
Bases: exceptions.Exception

exception ternip.formats.xml_doc.**TokeniseError**(*s*)
Bases: exceptions.Exception

class ternip.formats.xml_doc.**XmlDocument**(*file*, *nodename=None*, *has_S=False*, *has_LEX=False*, *pos_attr=False*)
Bases: object

An abstract base class which all XML types can inherit from. This implements almost everything, apart from the conversion of timex objects to and from timex tags in the XML. This is done by child classes

static create(*sents*, *tok_offsets=None*, *add_S=False*, *add_LEX=False*, *pos_attr=False*)
This is an abstract function for building XML documents from the internal representation only. You are not guaranteed to get out of get_sents what you put in here. Sentences and words will be retokenised and retagged unless you explicitly add S and LEX tags and the POS attribute to the document using the optional arguments.

sents is the [[(word, pos, timexes), ...], ...] format.

tok_offsets is used to correctly reinsert whitespace lost in tokenisation. It's in the format of a list of lists of integers, where each integer is the offset from the start of the sentence of that token. If set to None (the default), then a single space is assumed between all tokens.

If add_S is set to something other than false, then the tags to indicate sentence boundaries are added, with the name of the tag being the value of add_S

add_LEX is similar, but for token boundaries

pos_attr is similar but refers to the name of the attribute on the LEX (or whatever) tag that holds the POS tag.

get_dct_sents()
Returns the creation time sents for this document.

get_sents()
Returns a representation of this document in the [[(word, pos, timexes), ...], ...] format.

If there are any TIMEXes in the input document that cross sentence boundaries (and the input is not already broken up into sentences with the S tag), then those TIMEXes are disregarded.

reconcile(*sents*, *add_S=False*, *add_LEX=False*, *pos_attr=False*)
Reconciles this document against the new internal representation. If add_S is set to anything other than False, this means tags are indicated to indicate the sentence boundaries, with the tag names being the value of add_S. add_LEX is the same, but for marking token boundaries, and pos_attr is the name of the attribute which holds the POS tag for that token. This is mainly useful for transforming the TERN documents into something that GUTime can parse.

If your document already contains S and LEX tags, and add_S/add_LEX is set to add them, old S/LEX tags will be stripped first. If pos_attr is set and the attribute name differs from the old POS attribute name on the lex tag, then the old attribute will be removed.

Sentence/token boundaries will not be altered in the final document unless add_S/add_LEX is set. If you have changed the token boundaries in the internal representation from the original form, but are not then adding them back in, reconciliation may give undefined results.

There are some inputs which would output invalid XML. For example, if this document has elements which span multiple sentences, but not whole parts of them, then you will be unable to add XML tags and get valid XML, so failure will occur in unexpected ways.

If you are adding LEX tags, and your XML document contains tags internal to tokens, then reconciliation will fail, as it expects tokens to be in a continuous piece of whitespace.

reconcile_dct(*dct*, *add_S=False*, *add_LEX=False*, *pos_attr=False*)
Adds a TIMEX to the DCT tag and return the DCT

strip_tag(*tagname*)
Remove this tag from the document.

strip_timexes()
Strips all timexes from this document. Useful if we're evaluating the software - we can just feed in the gold standard directly and compare the output then.

rule_engine Package

expressions Module

normalisation_rule Module

class ternip.rule_engine.normalisation_rule.**NormalisationRule**(*match*, *type=None*, *id=''*, *value=None*, *change_type=None*, *freq=None*, *quant=None*, *mod=None*, *guards=None*, *after_guards=None*, *before_guards=None*, *sent_guards=None*, *after=None*, *tokenise=True*, *delimiate_numbers=False*)

Bases: *ternip.rule_engine.rule.Rule*

A class that represents normalisation rules

apply(*timex*, *cur_context*, *dct*, *body*, *before*, *after*)
Applies this rule to this timex, where body is the full extent covered by this timex, before is the preceeding text in the sentence, and after is the proceeding text in the sentence, in the [(token, POS), ...] form

A boolean indicating whether or not application was successful is returned. The timex may also be modified, so should be passed in by reference.

normalisation_rule_block Module

class ternip.rule_engine.normalisation_rule_block.**NormalisationRuleBlock**(*id*, *after*, *type*, *rules*)

Bases: *ternip.rule_engine.rule_block.RuleBlock*

A block of normalisation rules

apply(*timex*, *cur_context*, *dct*, *body*, *before*, *after*)

Apply rules in this block, in order, to this sentence, either until one rule is successful, or all rules have been applied.

normalisation_rule_engine Module

class ternip.rule_engine.normalisation_rule_engine.**NormalisationRuleEngine**

Bases: *ternip.rule_engine.rule_engine.RuleEngine*

A class which does normalisation using a rule engine

Complex rules must have a string member called 'id', which is used for after ordering, a list of strings called 'after' (which can be an empty list) which consists of IDs that must have run before this rule. Additionally, a function called 'apply' which takes a list of (token, pos, timexes) tuples and returns them in the same form with potentially modified timexes.

annotate(*sents*, *dct*)

This annotates all the timexes in the sents. dct means the document creation time (in the TIDES-modified ISO8601 format), which some rules may use to determine a context.

recognition_rule Module

class ternip.rule_engine.recognition_rule.**RecognitionRule**(*match*, *type*, *id*, *guards=None*, *after_guards=None*, *before_guards=None*, *after=None*, *squelch=False*, *case_sensitive=False*, *deliminate_numbers=False*)

Bases: *ternip.rule_engine.rule.Rule*

A class that represents identification rules

apply(*sent*)

Applies this rule to the tokenised sentence. The 'after' ordering must be checked by the caller to ensure correct rule application.

sent is a list of tuples (token, POS, [timexes])

A tuple is returned where the first element is a list in the same form as sent, with additional timexes added to the 3rd element if need be, and the second element in the tuple is whether or not this rule matched anything

recognition_rule_block Module

class ternip.rule_engine.recognition_rule_block.**RecognitionRuleBlock**(*id*, *after*, *type*, *rules*)

Bases: *ternip.rule_engine.rule_block.RuleBlock*

A block of recognition rules

apply(*sent*)

Apply rules in this block, in order, to this sentence, either until one rule is successful, or all rules have been applied.

recognition_rule_engine Module

class ternip.rule_engine.recognition_rule_engine.**RecognitionRuleEngine**

Bases: *ternip.rule_engine.rule_engine.RuleEngine*

A class which does recognition using a rule engine

Complex rules must have a string member called 'id', which is used for after ordering, a list of strings called 'after' (which can be an empty list) which consists of IDs that must have run before this rule. Additionally, a function called 'apply' which takes a list of (token, pos, timexes) tuples and returns them in the same form with potentially modified timexes.

tag(*sents*)

This function actually does word recognition. It expects content to be split into tokenised, POS tagged, sentences. i.e., a list of lists of tuples ([[(token, pos-tag, timexes), ...], ...]). Rules are applied one at a time.

What is returned is in the same form, except the token tuples contain a third element consisting of the set of timexes associated with that token.

rule Module

class ternip.rule_engine.rule.**Rule**

Bases: object

Base class for recognition and normalisation rules

rule_block Module

class ternip.rule_engine.rule_block.**RuleBlock**(*id*, *after*, *type*, *rules*)

Bases: object

rule_engine Module

class ternip.rule_engine.rule_engine.**RuleEngine**

Bases: object

A base class for rule engines to use

load_block(*filename*)

Load a block of rules, then check for consistency Throws rule_load_errors if a rule fails to load

load_rule(*filename*)
Load a rule, then check for consistency

Throws rule_load_error if a rule fails to load

load_rules(*path*)
Do rule loading. Loads all files ending in .pyrule as 'complex' rules (direct Python code), .rule using the documented rule format, and .ruleblock as blocks which contain sequences of rules. For direct Python code, the rule must be a class called 'rule'.

Throws rule_load_errors containing errors for all rules that failed to load.

exception ternip.rule_engine.rule_engine.**RuleLoadError**(*filename*, *errorstr*)
Bases: exceptions.Exception

Error for when a rule fails to load

exception ternip.rule_engine.rule_engine.**RuleLoadErrors**(*errors*)
Bases: exceptions.Exception

Error which bundles multiple rule_load_errors together. Allows for delayed exit on multiple load errors.

Subpackages

normalisation_functions Package

date_functions Module

ternip.rule_engine.normalisation_functions.date_functions.**convert_to_24_hours**(*time*, *ap*)
Given a hour and an a/p specifier, then convert the hour into 24 hour clock if need be

ternip.rule_engine.normalisation_functions.date_functions.**date_to_dow**(*y*, *m*, *d*)
Gets the integer day of week for a date. Sunday is 0.

ternip.rule_engine.normalisation_functions.date_functions.**date_to_iso**(*string*)
A translation of GUTime's Date2ISO function. Given some date/time string representing an absolute date, then return a date string in the basic ISO format.

ternip.rule_engine.normalisation_functions.date_functions.**date_to_week**(*y*, *m*, *d*)
Convert a date into a week number string, with year

ternip.rule_engine.normalisation_functions.date_functions.**easter_date**(*y*)
Return the date of Easter for that year as a string

ternip.rule_engine.normalisation_functions.date_functions.**extract_timezone**(*string*)
Given some string, try and extract the timezone it refers to. Returns a string.

ternip.rule_engine.normalisation_functions.date_functions.**normalise_two_digit_year**(*y*)
Given a year string, which could be 2 digits, try and get a 4 digit year out of it as a string

ternip.rule_engine.normalisation_functions.date_functions.**nth_dow_to_day**()
Figures out the day of the nth day-of-week in the month m and year y as an integer

e.g., 2nd Wednesday in July 2010: nth_dow_to_day((7, 3, 2), 2010)

Conversion from GUTime

relative_date_functions Module

ternip.rule_engine.normalisation_functions.relative_date_functions.**compute_offset_base**(*ref_d*
ex-
pres-
sion,
cur-
*rent_*

Given a reference date, some simple expression (yesterday/tomorrow or a day of week) and the direction of the relative expression, the base date with which to compute the offset from as a date string

ternip.rule_engine.normalisation_functions.relative_date_functions.**offset_from_date**(*v*,
off-
set,
gran='D'
ex-
act=False

Given a date string and some numeric offset, as well as a unit, then compute the offset from that value by offset gran's. Gran defaults to D. If exact is set to true, then the exact date is figured out, otherwise the level of granuality given by gran is used. Returns a date string.

ternip.rule_engine.normalisation_functions.relative_date_functions.**relative_direction_heur**

Given what preceeds and proceeds a TIMEX, then use heuristics to use tense to compute which direction a relative expression is in. Converted from GUTime.

string_conversions Module

Functions which convert strings to some number index

ternip.rule_engine.normalisation_functions.string_conversions.**build_duration_value**(*num*,
unit)

ternip.rule_engine.normalisation_functions.string_conversions.**day_to_num**(*day*)
Given the name of a day, the number of that day. Sunday is 0. Invalid data gets 7. All returned as integers.

ternip.rule_engine.normalisation_functions.string_conversions.**decade_nums**(*dec*)
Given the decade component (less the ty suffix) of a year, the number of that year as an integer.

ternip.rule_engine.normalisation_functions.string_conversions.**fixed_holiday_date**(*hol*)
Get the date string MMDD of a holiday

ternip.rule_engine.normalisation_functions.string_conversions.**month_to_num**(*m*)
Given a name of a month, get the number of that month. Invalid data gets 0. Returned as an integer.

ternip.rule_engine.normalisation_functions.string_conversions.**nth_dow_holiday_date**(*hol*)
Given the name of a holiday which always occur on the Nth X of some month, where X is day of week, returns tuples of the form (month, dow, n) representing the information about that holiday.

ternip.rule_engine.normalisation_functions.string_conversions.**season**(*s*)
Transforms a season name into an identifer from TIDES. Invalid data gets returned as is

ternip.rule_engine.normalisation_functions.string_conversions.**season_to_month**(*s*)
Convert seasons to months (roughly), returns an int

ternip.rule_engine.normalisation_functions.string_conversions.**units_to_gran**(*unit*)
Given a word, or part of a word, that represents a unit of time, return the single character representing the granuality of that unit of time

words_to_num Module

ternip.rule_engine.normalisation_functions.words_to_num.**ordinal_to_num**(*o*)
Given an ordinal (i.e., thirty-first or second) in the range 1st-31st (both numbers and words accepted), return the number value of that ordinal. Unrecognised data gets 1. Returns an integer

ternip.rule_engine.normalisation_functions.words_to_num.**words_to_num**(*words*)
Converted from GUTime. Given a string of number words, attempts to derive the numerical value of those words. Returns an integer.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

ternip.__init__, 3
ternip.formats, 3
ternip.formats.gate, 3
ternip.formats.tempeval2, 4
ternip.formats.tern, 4
ternip.formats.timeml, 5
ternip.formats.timex2, 5
ternip.formats.timex3, 5
ternip.formats.xml_doc, 6
ternip.rule_engine.expressions, 7
ternip.rule_engine.normalisation_functions.date_functions, 10
ternip.rule_engine.normalisation_functions.relative_date_functions, 11
ternip.rule_engine.normalisation_functions.string_conversions, 11
ternip.rule_engine.normalisation_functions.words_to_num, 12
ternip.rule_engine.normalisation_rule, 7
ternip.rule_engine.normalisation_rule_block, 8
ternip.rule_engine.normalisation_rule_engine, 8
ternip.rule_engine.recognition_rule, 8
ternip.rule_engine.recognition_rule_block, 9
ternip.rule_engine.recognition_rule_engine, 9
ternip.rule_engine.rule, 9
ternip.rule_engine.rule_block, 9
ternip.rule_engine.rule_engine, 9
ternip.timex, 3

Index

A

add_timex_ids() (in module ternip.timex), 3
annotate() (ternip.rule_engine.normalisation_rule_engine.NormalisationRuleEngine method), 8
apply() (ternip.rule_engine.normalisation_rule.NormalisationRule method), 7
apply() (ternip.rule_engine.normalisation_rule_block.NormalisationRuleBlock method), 8
apply() (ternip.rule_engine.recognition_rule.RecognitionRule method), 8
apply() (ternip.rule_engine.recognition_rule_block.RecognitionRuleBlock method), 9

B

BadNodeNameError, 6
build_duration_value() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11

C

compute_offset_base() (in module ternip.rule_engine.normalisation_functions.relative_date_functions), 11
convert_to_24_hours() (in module ternip.rule_engine.normalisation_functions.date_functions), 10
create() (ternip.formats.tempeval2.TempEval2Document static method), 4
create() (ternip.formats.tern.TernDocument static method), 4
create() (ternip.formats.timeml.TimeMlDocument static method), 5
create() (ternip.formats.xml_doc.XmlDocument static method), 6

D

date_to_dow() (in module ternip.rule_engine.normalisation_functions.date_functions), 10
date_to_iso() (in module ternip.rule_engine.normalisation_functions.date_functions), 10
date_to_week() (in module ternip.rule_engine.normalisation_functions.date_functions), 10
day_to_num() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11
decade_nums() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11

E

easter_date() (in module ternip.rule_engine.normalisation_functions.date_functions), 10
extract_timezone() (in module ternip.rule_engine.normalisation_functions.date_functions), 10

F

fixed_holiday_date() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11

G

GateDocument (class in ternip.formats.gate), 3
get_attrs() (ternip.formats.tempeval2.TempEval2Document method), 4
get_dct_sents() (ternip.formats.gate.GateDocument method), 4
get_dct_sents() (ternip.formats.tempeval2.TempEval2Document method), 4
get_dct_sents() (ternip.formats.tern.TernDocument method), 5
get_dct_sents() (ternip.formats.xml_doc.XmlDocument method), 6
get_extents() (ternip.formats.tempeval2.TempEval2Document method), 4

get_sents() (ternip.formats.gate.GateDocument method), 4
get_sents() (ternip.formats.tempeval2.TempEval2Document method), 4
get_sents() (ternip.formats.xml_doc.XmlDocument method), 6

L

load_block() (ternip.rule_engine.rule_engine.RuleEngine method), 9
load_multi() (ternip.formats.tempeval2.TempEval2Document static method), 4
load_rule() (ternip.rule_engine.rule_engine.RuleEngine method), 9
load_rules() (ternip.rule_engine.rule_engine.RuleEngine method), 10

M

month_to_num() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11

N

NestingError, 6
NormalisationRule (class in ternip.rule_engine.normalisation_rule), 7
NormalisationRuleBlock (class in ternip.rule_engine.normalisation_rule_block), 8
NormalisationRuleEngine (class in ternip.rule_engine.normalisation_rule_engine), 8
normalise_two_digit_year() (in module ternip.rule_engine.normalisation_functions.date_functions), 10
normaliser() (in module ternip.__init__), 3
nth_dow_holiday_date() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11
nth_dow_to_day() (in module ternip.rule_engine.normalisation_functions.date_functions), 10

O

offset_from_date() (in module ternip.rule_engine.normalisation_functions.relative_date_functions), 11
ordinal_to_num() (in module ternip.rule_engine.normalisation_functions.words_to_num), 12

R

recogniser() (in module ternip.__init__), 3
RecognitionRule (class in ternip.rule_engine.recognition_rule), 8
RecognitionRuleBlock (class in ternip.rule_engine.recognition_rule_block), 9
RecognitionRuleEngine (class in ternip.rule_engine.recognition_rule_engine), 9
reconcile() (ternip.formats.gate.GateDocument method), 4
reconcile() (ternip.formats.tempeval2.TempEval2Document method), 4
reconcile() (ternip.formats.xml_doc.XmlDocument method), 6
reconcile_dct() (ternip.formats.gate.GateDocument method), 4
reconcile_dct() (ternip.formats.tempeval2.TempEval2Document method), 4
reconcile_dct() (ternip.formats.tern.TernDocument method), 5
reconcile_dct() (ternip.formats.xml_doc.XmlDocument method), 7
relative_direction_heuristic() (in module ternip.rule_engine.normalisation_functions.relative_date_functio 11
Rule (class in ternip.rule_engine.rule), 9
RuleBlock (class in ternip.rule_engine.rule_block), 9
RuleEngine (class in ternip.rule_engine.rule_engine), 9
RuleLoadError, 10
RuleLoadErrors, 10

S

season() (in module ternip.rule_engine.normalisation_functions.string_conv 11
season_to_month() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11
strip_tag() (ternip.formats.xml_doc.XmlDocument method), 7
strip_timexes() (ternip.formats.xml_doc.XmlDocument method), 7

T

tag() (ternip.rule_engine.recognition_rule_engine.RecognitionRuleEngine method), 9
TempEval2Document (class in ternip.formats.tempeval2), 4
TernDocument (class in ternip.formats.tern), 4
ternip.__init__ (module), 3
ternip.formats (module), 3
ternip.formats.gate (module), 3
ternip.formats.tempeval2 (module), 4
ternip.formats.tern (module), 4
ternip.formats.timeml (module), 5
ternip.formats.timex2 (module), 5

ternip.formats.timex3 (module), 5
ternip.formats.xml_doc (module), 6
ternip.rule_engine.expressions (module), 7
ternip.rule_engine.normalisation_functions.date_functions (module), 10
ternip.rule_engine.normalisation_functions.relative_date_functions (module), 11
ternip.rule_engine.normalisation_functions.string_conversions (module), 11
ternip.rule_engine.normalisation_functions.words_to_num (module), 12
ternip.rule_engine.normalisation_rule (module), 7
ternip.rule_engine.normalisation_rule_block (module), 8
ternip.rule_engine.normalisation_rule_engine (module), 8
ternip.rule_engine.recognition_rule (module), 8
ternip.rule_engine.recognition_rule_block (module), 9
ternip.rule_engine.recognition_rule_engine (module), 9
ternip.rule_engine.rule (module), 9
ternip.rule_engine.rule_block (module), 9
ternip.rule_engine.rule_engine (module), 9
ternip.timex (module), 3
TimeMlDocument (class in ternip.formats.timeml), 5
Timex (class in ternip.timex), 3
Timex2XmlDocument (class in ternip.formats.timex2), 5
Timex3XmlDocument (class in ternip.formats.timex3), 5
TokeniseError, 6

U

units_to_gran() (in module ternip.rule_engine.normalisation_functions.string_conversions), 11

W

words_to_num() (in module ternip.rule_engine.normalisation_functions.words_to_num), 12

X

XmlDocument (class in ternip.formats.xml_doc), 6