
weblyzard_api Documentation

Release 0.1

Heinz-Peter Lang, Albert Weichselbraun

July 02, 2015

1	weblyzard_api Package	3
1.1	weblyzard_api Package	3
1.2	xml_content Module	3
1.3	client Module	7
2	data format	23
2.1	webLyzard XML Format	23
2.2	Annie-based Annotation Format	25
2.3	POS tags	27
2.4	List of Dependency Relations	30
2.5	Recognize	31
3	Indices and tables	35
	Python Module Index	37

Contents:

weblyzard_api Package

1.1 weblyzard_api Package

The webLyzard API package.

Provides support for webLyzard web services. Please refer to *client Module* for a list of available web services.

1.2 xml_content Module

Created on Feb, 27 2013

Handles the new (<http://www.weblyzard.com/wl/2013#>) weblyzard XML format.

Functions added:

- support for sentence tokens and pos iterators

Remove functions:

- compatibility fixes for namespaces, encodings etc.
- support for the old POS tags mapping.

class weblyzard_api.xml_content.**LabeledDependency** (*parent, pos, label*)

Bases: tuple

label

Alias for field number 2

parent

Alias for field number 0

pos

Alias for field number 1

class weblyzard_api.xml_content.**Sentence** (*md5sum=None, pos=None, sem_orient=None, significance=None, token=None, value=None, is_title=False, dependency=None*)

Bases: object

The sentence class used for accessing single sentences.

Note: the class provides convenient properties for accessing pos tags and tokens:

- s.sentence: sentence text
-

- `s.tokens` : provides a list of tokens (e.g. ['A', 'new', 'day'])
 - `s.pos_tags`: provides a list of pos tags (e.g. ['DET', 'CC', 'NN'])
-

as_dict()

Returns a dictionary representation of the sentence object.

dependency_list

Returns the dependencies of the sentence as a list of *LabeledDependency* objects

Return type `list` of `:py:class: weblyzard_api.xml_content.LabeledDependency` objects

```
>>> s = Sentence(pos='RB PRP MD', dependency='1:SUB -1:ROOT 1:OBJ')
>>> s.dependency_list
[LabeledDependency(parent='1', pos='RB', label='SUB'), LabeledDependency(parent='-1', pos='P
```

get_dependency_list()

Returns the dependencies of the sentence as a list of *LabeledDependency* objects

Return type `list` of `:py:class: weblyzard_api.xml_content.LabeledDependency` objects

```
>>> s = Sentence(pos='RB PRP MD', dependency='1:SUB -1:ROOT 1:OBJ')
>>> s.dependency_list
[LabeledDependency(parent='1', pos='RB', label='SUB'), LabeledDependency(parent='-1', pos='P
```

get_pos_tags()

Get the POS Tags as list.

```
>>> sentence = Sentence(pos = 'PRP ADV NN')
>>> sentence.get_pos_tags()
['PRP', 'ADV', 'NN']
```

get_pos_tags_list()

Returns list of the sentence's POS tags

```
>>> sentence = Sentence(pos = 'PRP ADV NN')
>>> sentence.get_pos_tags_list()
['PRP', 'ADV', 'NN']
```

get_pos_tags_string()

Returns String of the sentence's POS tags

```
>>> sentence = Sentence(pos = 'PRP ADV NN')
>>> sentence.get_pos_tags_string()
'PRP ADV NN'
```

get_sentence()

get_tokens()

Returns an iterator providing the sentence's tokens

pos_tag_string

Returns String of the sentence's POS tags

```
>>> sentence = Sentence(pos = 'PRP ADV NN')
>>> sentence.get_pos_tags_string()
'PRP ADV NN'
```


pos_tags

Get the POS Tags as list.

```
>>> sentence = Sentence(pos = 'PRP ADV NN')
>>> sentence.get_pos_tags()
['PRP', 'ADV', 'NN']
```

pos_tags_list

Returns list of the sentence's POS tags

```
>>> sentence = Sentence(pos = 'PRP ADV NN')
>>> sentence.get_pos_tags_list()
['PRP', 'ADV', 'NN']
```

sentence**set_dependency_list** (*dependencies*)

Takes a list of `weblyzard_api.xml_content.LabeledDependency`

Parameters **dependencies** (*list*) – The dependencies to set for this sentence.

Note: The list must contain items of the type `weblyzard_api.xml_content.LabeledDependency`

```
>>> s = Sentence(pos='RB PRP MD', dependency='1:SUB -1:ROOT 1:OBJ')
>>> s.dependency_list
[LabeledDependency(parent='1', pos='RB', label='SUB'), LabeledDependency(parent='-1', pos='P
>>> s.dependency_list = [LabeledDependency(parent='-1', pos='MD', label='ROOT'), ]
>>> s.dependency_list
[LabeledDependency(parent='-1', pos='MD', label='ROOT')]
```

set_pos_tags (*new_pos_tags*)**set_pos_tags_list** (*pos_tags_list*)**set_pos_tags_string** (*new_value*)**set_sentence** (*new_sentence*)**tokens**

Returns an iterator providing the sentence's tokens

class `weblyzard_api.xml_content.XMLContent` (*xml_content*, *remove_duplicates=True*)

Bases: `object`

SUPPORTED_XML_VERSIONS = {'deprecated': <class 'weblyzard_api.xml_content.parsers.xml_deprecated.XMLDeprec

add_attribute (*key*, *value*)**classmethod apply_dict_mapping** (*attributes*, *mapping=None*)

as_dict (*mapping=None*, *ignore_non_sentence=False*, *add_titles_to_sentences=False*)
convert the XML content to a dictionary.

Parameters

- **mapping** – an optional mapping by which to restrict/rename the returned dictionary
- **ignore_non_sentence** – if true, sentences without without POS tags are omitted from the result

content_id**content_type**

classmethod `convert` (*xml_content*, *target_version*)

`get_content_id()`

`get_content_type()`

`get_lang()`

`get_nilsimsa()`

`get_plain_text()`

Returns the plain text of the XML content

`get_sentences()`

classmethod `get_text` (*text*)

Returns the utf-8 encoded text

`get_title()`

get_xml_document (*header_fields*='all', *sentence_attributes*=('pos_tags', 'sem_orient', 'significance', 'md5sum', 'pos', 'token', 'dependency'), *xml_version*=2013)

Parameters

- **header_fields** – the header_fields to include
- **sentence_attributes** – sentence attributes to include
- **xml_version** – version of the webLyzard XML format to use (XML2005.VERSION, XML2013.VERSION)

Returns the XML representation of the webLyzard XML object

classmethod `get_xml_version` (*xml_content*)

lang

nilsimsa

classmethod `parse_xml_content` (*xml_content*, *remove_duplicates*=True)

plain_text

Returns the plain text of the XML content

sentences

title

update_attributes (*new_attributes*)

updates the existing attributes with new ones

update_sentences (*sentences*)

updates the values of the existing sentences. if the list of sentence object is empty, sentence_objects will be set to the new sentences.

Parameters **sentences** – list of Sentence objects

Warning: this function will not add new sentences

1.3 client Module

1.3.1 client Package

client Package

webLyzard web service clients

classifier Module

Created on Jan 16, 2013

```
class weblyzard_api.client.classifier.Classifier(url='http://localhost:8080', usr=None,
                                                pwd=None)
```

Bases: `eWRT.ws.rest.MultiRESTClient`

Classifier

Provides support for text classification.

Parameters

- **url** – URL of the jeremia web service
- **usr** – optional user name
- **pwd** – optional password

CLASSIFIER_WS_BASE_PATH = `‘/joseph/rest/’`

```
classify(classifier_profile, weblyzard_xml, search_agents=None, num_results=1)
```

Classify weblyzard XML documents based on the given classifier profile.

Parameters

- **classifier_profile** – the profile to use for classification (e.g. ‘COMET’, ‘MK’)
- **weblyzard_xml** – weblyzard_xml representation of the document to classify
- **search_agents** – an optional list of search agents (e.g. `[1, 2, 3]`)
- **num_results** – number of classes to return

Returns the classification result

```
hello_world()
```

Simple hello world test.

```
train(classifier_profile, weblyzard_xml, correct_category, incorrect_category=None, document_timestamp=None)
```

Trains (and corrects) the classifier’s knowledge base.

Parameters

- **classifier_profile** – the profile to use for classification (e.g. ‘COMET’, ‘MK’)
- **weblyzard_xml** – weblyzard_xml representation of the document to learn
- **correct_category** – the correct category for the document
- **incorrect_category** – optional information on the incorrect category returned for this document

- **document_timestamp** – an optional timestamp, specifying when the document has been classified (used for retraining temporal knowledge bases)

Returns a response object with a status code and message.

domain_specificity Module

```
class weblyzard_api.client.domain_specificity.DomainSpecificity(url='http://localhost:8080',  
                                                                usr=None,  
                                                                pwd=None)
```

Bases: `eWRT.ws.rest.MultiRESTClient`

Domain Specificity Web Service

Determines whether documents are relevant for a given domain by searching for domain relevant terms in these documents.

Workflow

1. submit a domain-specificity profile with `add_profile()`
2. obtain the domain-specificity of text documents with `get_domain_specificity()`, `parse_documents()` or `search_documents()`.

Parameters

- **url** – URL of the jeremia web service
- **usr** – optional user name
- **pwd** – optional password

URL_PATH = 'rest/domain_specificity'

add_profile (*profile_name*, *profile_mapping*)

Adds a domain-specificity profile to the Web service.

Parameters

- **profile_name** – the name of the domain specificity profile
- **profile_mapping** – a dictionary of keywords and their respective domain specificity values.

get_domain_specificity (*profile_name*, *documents*, *is_case_sensitive=True*)

Parameters

- **profile_name** – the name of the domain specificity profile to use.
- **documents** – a list of dictionaries containing the document
- **is_case_sensitive** – whether to consider case or not (default: True)

has_profile (*profile_name*)

Returns whether the given profile exists on the server.

Parameters **profile_name** – the name of the domain specificity profile to check.

Returns True if the given profile exists on the server.

list_profiles ()

Returns a list of all available domain specificity profiles.

meminfo()

Returns Information on the web service's memory consumption

parse_documents (*matview_name, documents, is_case_sensitive=False*)

Parameters

- **matview_name** – a comma separated list of matview_names to check for domain specificity.
- **documents** – a list of dictionaries containing the document
- **is_case_sensitive** – case sensitive or not

Returns dict (profilename: (content_id, dom_spec))

search_documents (*profile_name, documents, is_case_sensitive=False*)

jeremia Module

```
class weblyzard_api.client.jeremia.Jeremia (url='http://localhost:8080',      usr=None,
                                           pwd=None)
```

Bases: `eWRT.ws.rest.MultiRESTClient`

Jeremia Web Service

Pre-processes text documents and returns an annotated webLyzard XML document.

Blacklisting

Blacklisting is an optional service which removes sentences which occur multiple times in different documents from these documents. Examples for such sentences are document headers or footers.

The following functions handle sentence blacklisting:

- `clear_blacklist()`
- `get_blacklist()`
- `submit_document_blacklist()`
- `update_blacklist()`

Jeremia returns a [webLyzard XML document](#). The weblyzard_api provides the class `XMLContent` to process and manipulate the weblyzard XML documents.:

Note: Example usage

```
from weblyzard_api.client.recognize import Recognize
from pprint import pprint

docs = {'id': '192292',
        'title': 'The document title.',
        'body': 'This is the document text...',
        'format': 'text/html',
        'header': {}}

client = Jeremia()
result = client.submit_document(docs)
pprint(result)
```

Parameters

- **url** – URL of the jeremia web service
- **usr** – optional user name
- **pwd** – optional password

ATTRIBUTE_MAPPING = {'lang': 'lang', 'sentences_map': {'token': 'token', 'md5sum': 'id', 'pos': 'pos', 'value': 'value'}}

URL_PATH = 'jeremia/rest'

clear_blacklist (*source_id*)

Parameters **source_id** – the blacklist's source id

Empties the existing sentence blacklisting cache for the given *source_id*

commit (*batch_id*)

Parameters **batch_id** – the *batch_id* to retrieve

Returns a generator yielding all the documents of that particular batch

get_blacklist (*source_id*)

Parameters **source_id** – the blacklist's source id

Returns the sentence blacklist for the given *source_id*

get_xml_doc (*text*, *content_id*='1')

Processes text and returns a XMLContent object.

Parameters

- **text** – the text to process
- **content_id** – optional content id

status ()

Returns the status of the Jeremia web service.

submit (*batch_id*, *documents*, *source_id*=None, *use_blacklist*=False)

Convenience function to submit documents. The function will submit the list of documents and finally call commit to retrieve the result

Parameters

- **batch_id** – ID of the batch
- **documents** – list of documents (dict)
- **source_id** –
- **use_blacklist** – use the blacklist or not

Returns result as a list with dicts

submit_document (*document*)

processes a single document with jeremia (annotates a single document)

Parameters **document** – the document to be processed

submit_documents (*batch_id*, *documents*)

Parameters

- **batch_id** – *batch_id* to use for the given submission
- **documents** – a list of dictionaries containing the document

submit_documents_blacklist (*batch_id*, *documents*, *source_id*)

submits the documents and removes blacklist sentences

Parameters

- **batch_id** – batch_id to use for the given submission
- **documents** – a list of dictionaries containing the document
- **source_id** – source_id for the documents, determines the blacklist

update_blacklist (*source_id*, *blacklist*)

updates an existing blacklist cache

Parameters **source_id** – the blacklist's source id

version ()

Returns the current version of the jeremia deployed on the server

jesaja Module

class weblyzard_api.client.jesaja.**Jesaja** (*url='http://localhost:8080'*, *usr=None*, *pwd=None*)

Bases: eWRT.ws.rest.MultiRESTClient

Provides access to the Jesaja keyword service.

Jesaja extracts associations (i.e. keywords) from text documents.

Parameters

- **url** – URL of the jeremia web service
- **usr** – optional user name
- **pwd** – optional password

ATTRIBUTE_MAPPING = {'lang': 'xml:lang', 'sentences_map': {'token': 'token', 'md5sum': 'id', 'pos': 'pos', 'value': 'v

URL_PATH = 'jesaja/rest'

VALID_CORPUS_FORMATS = ('xml', 'csv')

add_or_update_corpus (*corpus_name*, *corpus_format*, *corpus*, *profile_name=None*,
skip_profile_check=False)

Adds/updates a corpus at Jesaja.

Parameters

- **corpus_name** – the name of the corpus
- **corpus_format** – either 'csv', 'xml', or wxml
- **corpus** – the corpus in the given format.
- **profile_name** – the name of the profile used for tokenization (only used in conjunction with corpus_format 'doc').

Note: Supported corpus_format

- csv
- xml
- wxml:

```
# xml_content: the content in the weblyzard xml format
corpus = [ xml_content, ... ]
```

Attention: uploading documents (corpus_format = doc, wlxml) requires a call to finalize_corpora to trigger the corpus generation!

add_or_update_stoplist (name, stoplist)

Deprecated since version 0.1: Use: `add_stoplist()` instead.

add_profile (profile_name, keyword_calculation_profile)

Add a keyword profile to the server

Parameters

- **profile_name** – the name of the keyword profile
- **keyword_calculation_profile** – the full keyword calculation profile (see below).

Note: Example keyword calculation profile

```
{
  'valid_pos_tags'          : ['NN', 'P', 'ADJ'],
  'corpus_name'             : reference_corpus_name,
  'min_phrase_significance' : 2.0,
  'num_keywords'           : 5,
  'keyword_algorithm'       : 'com.weblyzard.backend.jesaja.algorithm.keywords.YatesKeywordSignificanceAlgorithm',
  'min_token_count'        : 5,
  'skip_underrepresented_keywords' : True,
  'stoplists'              : [],
}
```

Note: Available keyword_algorithms

- `com.weblyzard.backend.jesaja.algorithm.keywords.YatesKeywordSignificanceAlgorithm`
- `com.weblyzard.backend.jesaja.algorithm.keywords.LogLikelihoodKeywordSignificanceAlgorithm`

add_stoplist (name, stoplist)

Parameters

- **name** – name of the stopword list
- **stoplist** – a list of stopwords for the keyword computation

change_log_level (level)

Changes the log level of the keyword service

Parameters **level** – the new log level to use.

classmethod convert_document (xml)

converts an XML String to a dictionary with the correct parameters (ignoring non-sentences and adding the titles)

Parameters **xml** – str representing the document

Returns converted document

Return type dict

finalize_corpora()

Note: This function needs to be called after uploading 'doc' or 'wxml' corpora, since it triggers the computations of the token counts based on the 'valid_pos_tags' parameter.

finalize_profile (*profile_name*)

get_cache_stats()

get_cached_corpora()

get_corpus_size (*profile_name*)

classmethod get_documents (*xml_content_dict*)

converts a list of weblyzard xml files to the json format required by the jesaja web service.

get_keywords (*profile_name, documents*)

Parameters

- **profile_name** – keyword profile to use
- **documents** – a list of webLyzard xml documents to annotate

Note: example documents list

```
documents = [
    {
        'title': 'Test document',
        'sentence': [
            {
                'id': '27150b5fae553ebab63332fe7b94d518',
                'pos': 'NNP VBZ VBN IN VBZ NNP . NNP VBZ NNP .',
                'token': '0,5 6,8 9,16 17,19 20,27 28,43 43,44 45,48 49,54 55,61 61,62',
                'value': 'CDATA is wrapped as follows <![CDATA[aha]]>. Ana loves Martin.'
            },
            {
                'id': 'f8ddd9b3c8cf4c7764a3348d14e84e79',
                'pos': 'NN IN CD ' IN JJR JJR JJR JJR CC CC : : JJ NN .',
                'token': '0,4 5,7 8,9 10,11 12,16 17,18 18,19 19,20 20,21 22,23 23,24 25,28 29,30',
                'value': '10µm in € " with <><> && and // related stuff.'
            }
        ],
        'content_id': '123k233',
        'lang': 'en',
    }
]
```

get_keywords_xml (*profile_name, documents*)

converts each document to a dictionary and calculates the keywords

has_profile (*profile_name*)

list_profiles()

list_stoplists()

Returns a list of all available stopword lists.

meminfo()

pos Module

Part-of-speech (POS) tagging service

```
class weblyzard_api.client.pos.POS (url='http://voyager.srv.weblyzard.net/ws',      usr=None,
                                   pwd=None)
    Bases: eWRT.ws.rest.RESTClient
```

Parameters

- **url** – URL of the jeremia web service
- **usr** – optional user name
- **pwd** – optional password

```
pos_tagging (text, lang)
```

tags the following text using the given language dictionary

Returns the corresponding ANNIE compatible annotations

recognize Module

output format

Overview The result of a call to recognize with multiple profiles (e.g. geoname, organizations, ...) returns a dictionary with keys being the respective entity names (GeoEntity, OrganizationEntity, PersonEntity).

Recognize supports three different output formats:

- **Standard**, returns one annotated result per found instance. Also returns all respective bindings specified in the profile.
- **Annie**, returns one annotated result per found instance. Returns all candidate groundings found in the GATE Annie format.
- **Compact**, returns one annotated result per found entity. Multiple matches of the same entity are returned as a single annotation with the individual spans saved as entities. The compact format is optimized for the Weblyzard use case.

Notice: Only the Annie and the Compact formats support sentence level annotation.

Annie

```
dict: {
  u'GeoEntity': [
    {
      u'confidence': 0.0,
      u'end': 0,
      u'features': {
        u'profile': u'en.geo.500000',
        u'entities': [
          {
            u'url': u'http://sws.geonames.org/5551752/',
            u'confidence': 0.0,
            u'preferredName': u'Arizona'
          }
        ]
      },
      u'grounded': False,
      u'sentence': 0,
```

```

        u'scoreName': u'GEO FOCUS x OCCURENCE',
        u'entityType': u'GeoEntity',
        u'start': 0,
        u'score': 2.57,
        u'profileName': u'en.geo.500000',
        u'preferredName': u'Arizona'
    },
    {
        ...
    }
]
}

```

Compact

```

{
    u'GeoEntity': [
        {
            u'confidence': 9.0,
            u'entities': [
                {
                    u'end': 7,
                    u'sentence': 15,
                    u'start': 0,
                    u'surfaceForm': u'Detroit'
                },
                {
                    u'end': 10,
                    u'sentence': 16,
                    u'start': 3,
                    u'surfaceForm': u'Detroit'
                }
            ],
            u'entityType': u'GeoEntity',
            u'key': u'http://sws.geonames.org/4990729/',
            u'preferredName': u'Detroit',
            u'profileName': u'en.geo.500000',
            u'properties': {
                u'adminLevel': u'http://www.geonames.org/ontology#P.PPLA2',
                u'latitude': u'42.33143',
                u'longitude': u'-83.04575',
                u'parent': u'http://sws.geonames.org/5014227/',
                u'parentCountry': u'http://sws.geonames.org/6252001/',
                u'population': u'713777'
            },
            u'score': 18.88
        },
        {
            ...
        }
    ],
    u'OrganizationEntity': [
        {
            u'confidence': 1277.1080389750275,
            u'entities': [{u'end': 101,
                u'sentence': 12,
                u'start': 87,
                u'surfaceForm': u'P

```

```
}
    u'entityType': u'OrganizationEntity',
    u'key': u'http://dbpedia.org/resource/Public_Service_Enterprise',
    u'preferredName': u'Public Service Enterprise',
    u'profileName': u'en.organization.ng',
    u'properties': {},
    u'score': 1277.11}]
```

Recognize Annotation Service

Welcome This is the public access point to the demo of the [Recognize Annotation Service](#). Given a text input, the Recognize service returns a set of [Named Entities](#) together with their start and end positions within the input text. Under the hood, Recognize makes use of open data portals such as [DBpedia](#) and [Geonames](#) for its queries, returning predefined subsets (property-wise) of respective entities.

Service usage is limited to 100 requests per day (max. 1MB data transfer per request)

Search Profiles When querying Recognize, you must provide a search profile to search within. A search profile describes a domain from the real world, and currently there exist the following set of domains:

{en,de}.organization.ng Organizations in english and german, taken from DBpedia. Returns type `OrganizationEntity`

{en,de}.people.ng Person Names in english and german, taken from DBpedia. Returns type `PersonEntity`

{en,de,fr}.geo.50000.ng Geolocations (cities, countries) with a population larger than 50000, taken from GeoNames. Returns type `GeoEntity`

Passing multiple profiles at once is also supported by the API.

REST API The REST interface can easily be accessed via our open source [weblyzard API](#) as shown below.

```
from weblyzard_api.client.recognize import Recognize
from pprint import pprint

url = 'http://triple-store.ai.wu.ac.at/recognize/rest/recognize'
profile_names=['en.organization.ng', 'en.people.ng', 'en.geo.50000.ng']
text = 'Microsoft is an American multinational corporation headquartered in Redmond, Washington, that'
client = Recognize(url)
result = client.search_text(profile_names,
                            text,
                            output_format='compact',
                            max_entities=40,
                            buckets=40,
                            limit=40)

pprint(result)
```

Results Recognize returns a JSON list object of all found entities. For each found entity, the service returns the entity type, the associated search profile (see above), the entity's occurrences within the given text (start, end, sentence, surface form), the confidence of the found entity to be correct, the public key where the entity links to (e.g. <http://sws.geonames.org/4990729>), as well as extra properties where available.

```
[{u'confidence': 6385.540194875138,
  u'entities': [{u'end': 22,
                  u'sentence': 0,
                  u'start': 1,
```

```

        u'surfaceForm': u'Microsoft Corporation'}}},
u'entityType': u'OrganizationEntity',
u'key': u'http://dbpedia.org/resource/Microsoft_Corporation',
u'preferredName': u'Microsoft Corporation',
u'profileName': u'en.organization.ng',
u'properties': {},
u'score': 6385.54},
{u'confidence': 4.0,
u'entities': [{u'end': 100,
                u'sentence': 0,
                u'start': 90,
                u'surfaceForm': u'Washington'}}},
u'entityType': u'GeoEntity',
u'key': u'http://sws.geonames.org/4140963/',
u'preferredName': u'Washington D.C.',
u'profileName': u'en.geo.500000.ng',
u'properties': {u'adminLevel': u'http://www.geonames.org/ontology#P.PPLC',
                u'latitude': u'38.89511',
                u'longitude': u'-77.03637',
                u'parent': u'http://sws.geonames.org/4138106/',
                u'parentCountry': u'http://sws.geonames.org/6252001/',
                u'population': u'601723'},
u'score': 3.15},
{u'confidence': 1808.274919947148,
u'entities': [{u'end': 269,
                u'sentence': 0,
                u'start': 259,
                u'surfaceForm': u'Bill Gates'}}},
u'entityType': u'PersonEntity',
u'key': u'http://dbpedia.org/resource/Bill_Gates',
u'preferredName': u'Bill Gates',
u'profileName': u'en.people.ng',
u'properties': {u'birthDate': u'1955-10-28',
                u'givenName': u'Bill',
                u's': u'http://dbpedia.org/resource/Bill_Gates',
                u'surname': u'Gates',
                u'thumbnail': u'http://upload.wikimedia.org/wikipedia/commons/4/4a/B'},
u'score': 1808.27},
{u'confidence': 1808.274919947148,
u'entities': [{u'end': 284,
                u'sentence': 0,
                u'start': 274,
                u'surfaceForm': u'Paul Allen'}}},
u'entityType': u'PersonEntity',
u'key': u'http://dbpedia.org/resource/Paul_Allen',
u'preferredName': u'Paul Allen',
u'profileName': u'en.people.ng',
u'properties': {u'birthDate': u'1953-01-21',
                u'givenName': u'Paul',
                u's': u'http://dbpedia.org/resource/Paul_Allen',
                u'surname': u'Allen',
                u'thumbnail': u'http://upload.wikimedia.org/wikipedia/commons/5/51/P'},
u'score': 1808.27}}]

```

class weblyzard_api.client.recognize.**EntityLyzardTest** (*methodName='runTest'*)

Bases: unittest.case.TestCase

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

```
DOCS = [{'xml:lang': 'de', 'id': 99933, 'sentence': [{'token': '0,5 6,12 13,16 17,19 20,23 24,27 28,36 36,37', 'id': '50612085'}]}]
DOCS_XML = '\n <?xml version="1.0" encoding="UTF-8"?>\n <wl:page xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:wl="http://purl.org/dc/elements/1.1/">\n </wl:page>\n </xml>'
IS_ONLINE = True
TESTED_PROFILES = ['de.people.ng', 'en.geo.500000.ng', 'en.organization.ng', 'en.people.ng']

setUp()
test_entity_lyzard()
test_focus_search()
test_geo()
test_missing_profiles()
test_organization()
test_password()
test_people()
test_search_xml()

xml = '\n <?xml version="1.0" encoding="UTF-8"?>\n <wl:page xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:wl="http://purl.org/dc/elements/1.1/">\n </wl:page>\n </xml>'

class webyzard_api.client.recognize.Recognize(url='http://localhost:8080',      usr=None,
                                             pwd=None)
    Bases: eWRT.ws.rest.MultiRESTClient
    Provides access to the Recognize Web Service.
```

Workflow:

1. pre-load the recognize profiles you need using the `add_profile()` call.
2. submit the text or documents to analyze using one of the following calls:
 - `search_document()` or `search_documents()` for document dictionaries.
 - `search_text()` for plain text.

Note: Example usage

```
from webyzard_api.client.recognize import Recognize
from pprint import pprint

url = 'http://triple-store.ai.wu.ac.at/recognize/rest/recognize'
profile_names = ['en.organization.ng', 'en.people.ng', 'en.geo.500000.ng']
text = 'Microsoft is an American multinational corporation
        headquartered in Redmond, Washington, that develops,
        manufactures, licenses, supports and sells computer
        software, consumer electronics and personal computers
        and services. It was founded by Bill Gates and Paul
        Allen on April 4, 1975.'

client = Recognize(url)
result = client.search_text(profile_names,
                           text,
                           output_format='compact',
                           max_entities=40,
                           buckets=40,
                           limit=40)
pprint(result)
```

Parameters

- **url** – URL of the jeremia web service
- **usr** – optional user name
- **pwd** – optional password

ATTRIBUTE_MAPPING = {'lang': 'xml:lang', 'sentences_map': {'token': 'token', 'md5sum': 'id', 'pos': 'pos', 'value': 'v'}}

OUTPUT_FORMATS = ('standard', 'minimal', 'annie', 'compact')

URL_PATH = 'recognize/rest/recognize'

add_profile (*profile_name*, *force=False*)
pre-loads the given profile

::param *profile_name*: name of the profile to load.

classmethod convert_document (*xml*)
converts an XML String to a document dictionary necessary for transmitting the document to Recognize.

Parameters **xml** – weblyzard_xml representation of the document

Returns the converted document

Return type dict

Note: non-sentences are ignored and titles are added based on the XmlContent's interpretation of the document.

get_focus (*profile_names*, *doc_list*, *max_results=1*)

Parameters

- **profile_names** – a list of profile names
- **doc_list** – a list of documents to analyze based on the weblyzardXML format
- **max_results** – maximum number of results to include

Returns the focus and annotation of the given document

Note: Corresponding web call

<http://localhost:8080/recognize/focus?profiles=ofwi.people&profiles=ofwi.organizations.context>

get_xml_document (*document*)

Returns the correct XML representation required by the Recognize service

list_configured_profiles ()

Returns a list of all profiles supported in the current configuration

list_profiles ()

Returns a list of all pre-loaded profiles

```
>>> r=Recognize()
>>> r.list_profiles()
[u'Cities.DACH.10000.de_en', u'People.DACH.de']
```

remove_profile (*profile_name*)

removes a profile from the list of pre-loaded profiles

search_document (*profile_names*, *document*, *debug=False*, *max_entities=1*, *buckets=1*, *limit=1*, *output_format='minimal'*)

Parameters

- **profile_names** – a list of profile names
- **document** – a single document to analyze (see example documents below)
- **debug** – compute and return an explanation
- **buckets** – only return n buckets of hits with the same score
- **max_entities** – number of results to return (removes the top hit's tokens and rescores the result list subsequently)
- **limit** – only return that many results
- **output_format** – the output format to use ('standard', 'minimal', 'annie')

Return type the tagged dictionary

Note: Example document

```
# option 1: document dictionary
{'content_id': 12,
 'content': u'the text to analyze'}

# option 2: weblyzardXML
XMLContent('<?xml version="1.0"...').as_list()
```

Note: Corresponding web call

<http://localhost:8080/recognize/searchXml/ofwi.people>

search_documents (*profile_names*, *doc_list*, *debug=False*, *max_entities=1*, *buckets=1*, *limit=1*, *output_format='annie'*)

Parameters

- **profile_names** – a list of profile names
- **doc_list** – a list of documents to analyze (see example below)
- **debug** – compute and return an explanation
- **buckets** – only return n buckets of hits with the same score
- **max_entities** – number of results to return (removes the top hit's tokens and rescores the result list subsequently)
- **limit** – only return that many results
- **output_format** – the output format to use ('standard', 'minimal', 'annie')

Return type the tagged dictionary

Note: Example document


```
# option 1: list of document dictionaries
( {'content_id': 12,
  'content': u'the text to analyze'})

# option 2: list of weblyzardXML dictionary representations
(XMLContent('<?xml version="1.0"...').as_list(),
 XMLContent('<?xml version="1.0"...').as_list(),)
```

search_text (*profile_names*, *text*, *debug=False*, *max_entities=1*, *buckets=1*, *limit=1*, *output_format='minimal'*)

Search text for entities specified in the given profiles.

Parameters

- **profile_names** – the profile to search in
- **text** – the text to search in
- **debug** – compute and return an explanation
- **buckets** – only return n buckets of hits with the same score
- **max_entities** – number of results to return (removes the top hit's tokens and rescores the result list subsequently)
- **limit** – only return that many results
- **output_format** – the output format to use ('standard', 'minimal', 'annie')

Return type the tagged text

status ()

Returns the status of the Recognize web service.

sentiment_analysis Module

class weblyzard_api.client.sentiment_analysis.**SentimentAnalysis** (*url='http://voyager.srv.weblyzard.net/ws'*,
usr=None,
pwd=None)

Bases: eWRT.ws.rest.RESTClient

Sentiment Analysis Web Service

Parameters

- **url** – URL of the jeremia web service
- **usr** – optional user name
- **pwd** – optional password

parse_document (*text*, *lang*)

Returns the sentiment of the given text for the given language.

Parameters

- **text** – the input text
- **lang** – the text's language

Returns

sv; n_pos_terms; n_neg_terms; list of tuples, where each tuple contains two dicts:

- `tuple[0]`: ambiguous terms and its sentiment value after disambiguation
- `tuple[1]`: the context terms with their number of occurrences in the document.

parse_document_list (*document_list*, *lang*)

Returns the sentiment of the given text for the given language.

Parameters

- **document_list** – the input text
- **lang** – the text's language

Returns

sv; n_pos_terms; n_neg_terms; list of tuples, where each tuple contains two dicts:

- `tuple[0]`: ambiguous terms and its sentiment value after disambiguation
- `tuple[1]`: the context terms with their number of occurrences in the document.

reset (*lang*)

Restores the default data files for the given language (if available).

Parameters **lang** – the used language

Note: Currently this operation is only supported for German and English.

update_context (*context_dict*, *lang*)

Uploads the given context dictionary to the Web service.

Parameters

- **context_dict** – a dictionary containing the context information
- **lang** – the used language

update_lexicon (*corpus_dict*, *lang*)

Uploads the given corpus dictionary to the Web service.

Parameters

- **corpus_dict** – a dictionary containing the corpus information
- **lang** – the used language

update_negation (*negation_trigger_dict*, *lang*)

Uploads the given negation triggers to the Web service.

Parameters

- **negation_trigger_list** – a list of negation triggers to use with the given language
example: {'doesn't': 'doesnt',}
- **lang** – the used language

2.1 webLyzard XML Format

The webLyzard XML format is generated by the Jeremia Web service and encodes the following information:

- the document language (language detection)
- the document's nilsimsha hash (locality sensitive hashing)
- the document's input format (dc:format), creator (dc:creator), related named entities (dc:related)
- a list of all sentences used in the document (sentence splitting) including
 - the sentence's MD5 sum
 - an indication of whether the sentence is part of the document title
 - sentence tokens
 - part-of-speech tags (part of speech tagging)
 - dependencies (dependency parsing)
 - semantic orientation (text sentiment)
 - the sentence's significance (for the given domain)

Example webLyzard XML file:

```
<?xml version="1.0" encoding="utf8" ?>
<wl:page xmlns:wl="http://www.weblyzard.com/wl/2013#" xmlns:dc="http://purl.org/dc/elements/1.1/" wl:
  dc:format="text/html"
  dc:coverage="http://de.dbpedia.org/page/Helmut_Sch%C3%BCller
              http://de.dbpedia.org/page/Gerda_Schaffelhofer
              http://de.dbpedia.org/page/Styria_Media_Group"
  dc:creator="http://www.nachrichten.at/KA"
  dc:related="http://www.kurier.at/article/Die-Kirche.html http://www.diepresse.com/kirche/Katholische
  xml:lang="de"
  wl:nilsimsha="77799a10d691a16416300ae1fad0bbe24c3f0991c17533649db7cbe1e23d5241">

<!-- The title -->
<wl:sentence wl:id="61e8b085944f173e36637e8daf7d77c0"
  wl:token = "0:3 4:12 13:19 20:22 23:26 27:46 ...."
  wl:pos = "ADJA NN $. NN VVFIN NN XY"
  wl:dependency = "1 2 -1 4 5 2 2"
  wl:is_title = "True"
  wl:sem_orient="0.764719112902"
```

```
    wl:significance="None">
      <![CDATA[Katholische Aktion: Präsidentin rügt Pfarrer-Initiative | Nachrichten.at.]]>
</wl:sentence>

<!-- The content (wl:is_title = "False" or not set) -->
<wl:sentence wl:id="61e8b085944f173e36637e8daf7d77c0"
  wl:pos="APPR ADJA NN APPR ART NN APPR NE NE VVFIN APPRART NN ART ADJA NN ART ADJA NN NE (
  wl:dependency="1 -1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 1 1 1 1 1"
  wl:token="0,3 4,12 13,19 20,22 23,26 27,45 46,48 49,55 56,64 65,76 77,79 80,88 89,92 93,9
  wl:sem_orient="0.764719112902"
  wl:significance="None">
      <![CDATA[Mit scharfer Kritik an der Pfarrer-Initiative um Helmut Schüller überrascht
</wl:sentence>
<wl:sentence wl:id="a3cba1f907f41160e690ad072dd2fc08" wl:pos="NN ADJD APPR ART NN PPOSAT NN APPR ART
  <wl:sentence wl:id="3f49d4fe7e9fc31b74a8748e21002e23" wl:pos="NN NN KOUS ART NN PPOSAT NN APPR AR
</wl:page>
```

2.1.1 Dependency trees:

- `wl:dependency` describes the sentences' dependency structure. It consists of an integer and a string, concatenated by a column: `int:str`. The number refers to the current token's parent in the dependency tree. The string is the label/type of the dependency relationship, e.g. `nsubjpass`.
- Special values:
 - `-1`: root node
 - `-2`: no parent could be determined
- Example:
 - Text: Ana loves Tom, `wl:dependency: 1:SBJ -1:ROOT 1:OBJ`
 - Tree: "Anna -> loves <- Tom"

2.1.2 Changelog

- use <http://www.weblyzard.com/wl/2013#> as default namespace
- **include the dublin core namespace**
 - use `dc:title` rather than `title`
 - use `dc:format` rather than `content_type`
 - **Support the following constructs:**
 - * use `dc:creator` to refer to authors
 - * use `dc:coverage` to refer to companies, organizations and locations covered in the article
 - * multiple entries are separated by spaces
 - **Field names for document objects**
 - * `content_id` -> `wl:id`
 - * `content_type` -> `dc:format`
- 22 August 2014: add `wl:dependency` for dependency tries.
- **Python related changes**

- Document object {'content_id': 12, 'content_type': 'text/html' ... }
-> {'id': 12, 'format': 'text/html', ...}
 - Justification - wlid is required, i.e. the use of a proper namespace; the use of xml:id is not possible, because the XML Schema specification requires its values to be from type NCName (which does not allow values to start with a number!). - dc:format is a standardized identifier for the content type
- **26 January 2015:** Changed dependency format to include dependency labels.

2.2 Annie-based Annotation Format

The webLyzard/WISDOM annotation format is based on the data structures used by the [GATE](#) project. A detailed description of these data structures can be found in the [Gate Documentation on Language Resources: Corpora, Documents and Annotations](#).

2.2.1 Classes

- Annotation Set(type:String) - an Annotation Set contains “n” Annotations
- Annotation(start:int, end:int, type:String, feature=Map<String, String>)

2.2.2 Sentence-level annotations

Running example:

```
Andreas Wieland, CEO, Hamilton Bonaduz AG said: «We are very excited ...
01234567890123456789012345678901234567890123456789012345678901
0.....1.....2.....3.....4.....5.....6.....7.
```

*** Definition of the used JSON Fields ***
 * sentence: the sentence’s MD5 sum
 * start: the annotation’s start position within the sentence
 * end: the annotation’s end position within the sentence
 * type: the annotation type
 * features: a dictionary of annotation features

Geonames

```
[{
  "start":31,
  "end":38,
  "sentence": "777081b7ebe4a99b598ac2384483b4ab",
  "type": "ch.htwchur.wisdom.entityLyzard.GeoEntity",
  "features":{
    "entities":[{
      "confidence":7.0,
      "url": "http://sws.geonames.org/2661453/",
      "preferredName": "Bonaduz"
    }, {
      "confidence":6.0,
      "url": "http://sws.geonames.org/7285286/",
      "preferredName": "Bonaduz"
    }],
    "profile": "Cities.CH.de"
  }
}]
```

People

```
[{
  "start":0,
  "end":15,
  "sentence": "777081b7ebe4a99b598ac2384483b4ab",
  "type":"ch.htwchur.wisdom.entityLyzard.PersonEntity",
  "features":{"
    "entities":[{"
      "confidence":1646.4685797722482,
      "url":"http://www.semanticlab.net/proj/wisdom/ofwi/person/Andreas_Wieland_(0",
      "preferredName":"Andreas Wieland"
    },{
      "confidence":2214.9741075564775,
      "url":"http://www.semanticlab.net/proj/wisdom/ofwi/person/Andreas_Wieland_(0",
      "preferredName":"Andreas Wieland"
    },{
      "confidence":1646.4685797722482,
      "url":"http://www.semanticlab.net/proj/wisdom/ofwi/person/Andreas_Wieland_(0",
      "preferredName":"Andreas Wieland"
    },{
      "confidence":1646.4685797722482,
      "url":"http://www.semanticlab.net/proj/wisdom/ofwi/person/Andreas_Wieland_(0",
      "preferredName":"Andreas Wieland"
    },{
      "confidence":2165.3683447585117,
      "url":"http://www.semanticlab.net/proj/wisdom/ofwi/person/Andreas_Wieland_(0",
      "preferredName":"Andreas Wieland"
    }
  ]},
  "profile":"ofwi.people"
}
```

Organizations

```
[{
  "start":22,
  "end":41,
  "sentence": "777081b7ebe4a99b598ac2384483b4ab",
  "type":"ch.htwchur.wisdom.entityLyzard.OrganizationEntity",
  "features":{"
    "entities":[{"
      "confidence":438.9253911579335,
      "url":"http://www.semanticlab.net/proj/wisdom/ofwi/teledata/company/7246",
      "preferredName":"Hamilton Bonaduz AG"
    }
  ]},
  "profile":"ofwi.organizations"
}
```

Part-of-speech Tags

Please refer to used part-of-speech (POS) tags for a list of the POS-Tags used within webLyzard.

```
Anna is a student.
012345678901234567
0.....1.....
```

```
[
  {sentence="fbb1a44c0d422e496d87c3c8d23b4480", start=0, end=3, type="Token", features={ 'POS': 'NN' },
  {sentence="fbb1a44c0d422e496d87c3c8d23b4480", start=5, end=6, type="Token", features={ 'POS': 'VRB' },
  {sentence="fbb1a44c0d422e496d87c3c8d23b4480", start=8, end=8, type="Token", features={ 'POS': 'AR' },
  ...
]
```

2.3 POS tags

This file contains the used part-of-speech (POS)-tagsets for English, French and German. All used tags can also be found in [usedPosTags.csv](#) .

2.3.1 English

The English tagger uses the Penn Treebank POS tag set.

```
1. CC Coordinating conjunction
2. CD Cardinal number
3. DT Determiner
4. EX Existential there
5. FW Foreign word
6. IN Preposition or subordinating conjunction
7. JJ Adjective
8. JJR Adjective, comparative
9. JJS Adjective, superlative
10. LS List item marker
11. MD Modal
12. NN Noun, singular or mass
13. NNS Noun, plural
14. NNP Proper noun, singular
15. NNPS Proper noun, plural
16. PDT Predeterminer
17. POS Possessive ending
18. PRP Personal pronoun
19. PRP$ Possessive pronoun
20. RB Adverb
21. RBR Adverb, comparative
22. RBS Adverb, superlative
23. RP Particle
24. SYM Symbol
25. TO to
26. UH Interjection
27. VB Verb, base form
28. VBD Verb, past tense
29. VBG Verb, gerund or present participle
30. VBN Verb, past participle
31. VBP Verb, non-3rd person singular present
32. VBZ Verb, 3rd person singular present
33. WDT Wh-determiner
34. WP Wh-pronoun
```

35.	WP\$	Possessive wh-pronoun
36.	WRB	Wh-adverb

2.3.2 French

The French Tagger has been trained with the French Treebank corpus.

```
A (adjective)
Adv (adverb)
CC (coordinating conjunction)
Cl (weak clitic pronoun)
CS (subordinating conjunction)
D (determiner)
ET (foreign word)
I (interjection)
NC (common noun)
NP (proper noun)
P (preposition)
PREF (prefix)
PRO (strong pronoun)
V (verb)
PONCT (punctuation mark)
```

2.3.3 German

We use the Stuttgart-Tübingen-Tagset (STTS) that is used for the NEGRA Corpus.

ADJA	attributives Adjektiv	[das] große [Haus]
ADJD	adverbiales oder prädikatives Adjektiv	[er fährt] schnell [er ist] schnell
ADV	Adverb	schon, bald, doch
APPR	Präposition; Zirkumposition links	in [der Stadt], ohne [mich]
APPRART	Präposition mit Artikel	im [Haus], zur [Sache]
APPO	Postposition	[ihm] zufolge, [der Sache] wegen
APZR	Zirkumposition rechts	[von jetzt] an
ART	bestimmter oder unbestimmter Artikel	der, die, das, ein, eine, ...
CARD	Kardinalzahl	zwei [Männer], [im Jahre] 1994
FM	Fremdsprachliches Material	[Er hat das mit ``]
ITJ	Interjektion	mhm, ach, tja
ORD	Ordinalzahl	[der] neunte [August]
KOUI	unterordnende Konjunktion mit ``zu'' und Infinitiv	um [zu leben], anstatt [zu fragen]
KOUS	unterordnende Konjunktion mit Satz	weil, daß, damit, wenn, ob
KON	nebenordnende Konjunktion	und, oder, aber
KOKOM	Vergleichskonjunktion	als, wie

NN	normales Nomen	Tisch, Herr, [das] Reisen
NE	Eigennamen	Hans, Hamburg, HSV
PDS	substituierendes Demonstrativpronomen	dieser, jener
PDAT	attribuierendes Demonstrativpronomen	jener [Mensch]
PIS	substituierendes Indefinitpronomen	keiner, viele, man, niemand
PIAT	attribuierendes Indefinitpronomen ohne Determiner	kein [Mensch], irgendein [Glas]
PIDAT	attribuierendes Indefinitpronomen mit Determiner	[ein] wenig [Wasser], [die] beiden [Brüder]
PPER	irreflexives Personalpronomen	ich, er, ihm, mich, dir
PPOSS	substituierendes Possessivpronomen	meins, deiner
PPOSAT	attribuierendes Possessivpronomen	mein [Buch], deine [Mutter]
PRELS	substituierendes Relativpronomen	[der Hund ,] der
PRELAT	attribuierendes Relativpronomen	[der Mann ,] dessen [Hund]
PRF	reflexives Personalpronomen	sich, einander, dich, mir
PWS	substituierendes Interrogativpronomen	wer, was
PWAT	attribuierendes Interrogativpronomen	welche [Farbe], wessen [Hut]
PWAV	adverbiales Interrogativ- oder Relativpronomen	warum, wo, wann, worüber, wobei
PAV	Pronominaladverb	dafür, dabei, deswegen, trotzdem
PTKZU	``zu'' vor Infinitiv	zu [gehen]
PTKNEG	Negationspartikel	nicht
PTKVZ	abgetrennter Verbzusatz	[er kommt] an, [er fährt] rad
PTKANT	Antwortpartikel	ja, nein, danke, bitte
PTKA	Partikel bei Adjektiv oder Adverb	am [schönsten], zu [schnell]
SGML	SGML Markup	
SPELL	Buchstabierfolge	S-C-H-W-E-I-K-L
TRUNC	Kompositions-Erstglied	An- [und Abreise]
VVFIN	finites Verb, voll	[du] gehst, [wir] kommen [an]
VVIMP	Imperativ, voll	komm [!]
VVINFIN	Infinitiv, voll	gehen, ankommen
VVIZU	Infinitiv mit ``zu'', voll	anzukommen, loszulassen
VVPP	Partizip Perfekt, voll	gegangen, angekommen
VAFIN	finites Verb, aux	[du] bist, [wir] werden
VAIMP	Imperativ, aux	sei [ruhig !]
VAINF	Infinitiv, aux	werden, sein
VAPP	Partizip Perfekt, aux	gewesen
VMFIN	finites Verb, modal	dürfen

VMINF	Infinitiv, modal	wollen
VMPP	Partizip Perfekt, modal	gekonnt, [er hat gehen] können
XY	Nichtwort, Sonderzeichen enthaltend	3:7, H2O, D2XW3
\\$,	Komma	,
\\$.	Satzbeendende Interpunktion	. ? ! ; :
\\$(sonstige Satzzeichen; satzintern	- [,] ()

2.4 List of Dependency Relations

This file contains the used dependency relation tagset for English.

Source: “Dependency Syntax in the CoNLL Shared Task 2008” <http://wacky.sslmit.unibo.it/lib/exe/fetch.php?media=papers:conll-syntax.pdf>

1. ADV Unclassified adverbial
2. AMOD Modifier of adjective or adverb
3. APPO Apposition
4. BNF Benefactor (the for phrase for verbs that undergo dative shift)
5. CONJ Between conjunction and second conjunct in a coordination
6. COORD Coordination
7. DEP Unclassified relation
8. DIR Direction
9. DTV Dative (the to phrase for verbs that undergo dative shift)
10. EXT Extent
11. EXTR Extraposed element in expletive constructions
12. GAP Gapping: between conjunction and the parts of a structure with an ellipsed head
13. HMOD Modifier in hyphenation, such as two in two-part
14. HYPH Between first part of hyphenation and hyphen
15. IM Between infinitive marker and verb
16. LGS Logical subject
17. LOC Location
18. MNR Manner
19. NAME Name-internal link
20. NMOD Modifier of nominal
21. OBJ Direct or indirect object or clause complement
22. OPRD Object complement
23. P Punctuation
24. PMOD Between preposition and its child in a PP

25. POSTHON Posthonorifics such as Jr, Inc.
26. PRD Predicative complement
27. PRN Parenthetical
28. PRP Purpose or reason
29. PRT Particle
30. PUT Various locative complements of the verb put
31. ROOT Root
32. SBJ Subject
33. SUB Between subordinating conjunction and verb
34. SUFFIX Possessive 's
35. TITLE Titles such as Mr, Dr
36. TMP Temporal
37. VC Verb chain
38. VOC Vocative

2.5 Recognize

2.5.1 Overview

The result of a call to recognize with multiple profiles (e.g. geoname, organizations, ...) returns a dictionary with keys being the respective entity names (GeoEntity, OrganizationEntity, PersonEntity).

Recognize supports three different output formats:

- **Standard**, returns one annotated result per found instance. Also returns all respective bindings specified in the profile.
- **Annie**, returns one annotated result per found instance. Returns all candidate groundings found in the GATE Annie format.
- **Compact**, returns one annotated result per found entity. Multiple matches of the same entity are returned as a single annotation with the individual spans saved as entities. The compact format is optimized for the Weblyzard use case.

Note: Only the Annie and the Compact formats support sentence level annotation.

2.5.2 Annie

```
dict: {
  u'GeoEntity': [
    {
      u'confidence': 0.0,
      u'end': 0,
      u'features': {
        u'profile': u'en.geo.500000',
        u'entities': [
```

```
        {
            u'url': u'http://sws.geonames.org/5551752/',
            u'confidence': 0.0,
            u'preferredName': u'Arizona'}
    ],
    u'grounded': False,
    u'sentence': 0,
    u'scoreName': u'GEO FOCUS x OCCURENCE',
    u'entityType': u'GeoEntity',
    u'start': 0,
    u'score': 2.57,
    u'profileName': u'en.geo.500000',
    u'preferredName': u'Arizona'
},
{
    ...
}
]
```

2.5.3 Compact

```
{
  u'GeoEntity': [
    {
      u'confidence': 9.0,
      u'entities': [
        {
          u'end': 7,
          u'sentence': 15,
          u'start': 0,
          u'surfaceForm': u'Detroit'
        },
        {
          u'end': 10,
          u'sentence': 16,
          u'start': 3,
          u'surfaceForm': u'Detroit'
        }
      ],
      u'entityType': u'GeoEntity',
      u'key': u'http://sws.geonames.org/4990729/',
      u'preferredName': u'Detroit',
      u'profileName': u'en.geo.500000',
      u'properties': {
        u'adminLevel': u'http://www.geonames.org/ontology#P.PPLA2',
        u'latitude': u'42.33143',
        u'longitude': u'-83.04575',
        u'parent': u'http://sws.geonames.org/5014227/',
        u'parentCountry': u'http://sws.geonames.org/6252001/',
        u'population': u'713777'
      },
      u'score': 18.88
    },
    {
      ...
    }
  ]
}
```

```
    }
  ],
  u'OrganizationEntity': [
    {
      u'confidence': 1277.1080389750275,
      u'entities': [{u'end': 101,
        u'sentence': 12,
        u'start': 87,
        u'surfaceForm': u'Public Service'}]},
      u'entityType': u'OrganizationEntity',
      u'key': u'http://dbpedia.org/resource/Public_Service_Enterprise_Group',
      u'preferredName': u'Public Service Enterprise',
      u'profileName': u'en.organization.ng',
      u'properties': {},
      u'score': 1277.11}]
  ]
}
```

Indices and tables

- `genindex`
- `modindex`
- `search`

W

`weblyzard_api`, [3](#)
`weblyzard_api.client`, [7](#)
`weblyzard_api.client.classifier`, [7](#)
`weblyzard_api.client.domain_specificity`,
 [8](#)
`weblyzard_api.client.jeremia`, [9](#)
`weblyzard_api.client.jesaja`, [11](#)
`weblyzard_api.client.pos`, [14](#)
`weblyzard_api.client.recognize`, [17](#)
`weblyzard_api.client.sentiment_analysis`,
 [21](#)
`weblyzard_api.xml_content`, [3](#)

A

add_attribute() (weblyzard_api.xml_content.XMLContent method), 5
 add_or_update_corpus() (weblyzard_api.client.jesaja.Jesaja method), 11
 add_or_update_stoplist() (weblyzard_api.client.jesaja.Jesaja method), 12
 add_profile() (weblyzard_api.client.domain_specificity.DomainSpecificity method), 8
 add_profile() (weblyzard_api.client.jesaja.Jesaja method), 12
 add_profile() (weblyzard_api.client.recognize.Recognize method), 19
 add_stoplist() (weblyzard_api.client.jesaja.Jesaja method), 12
 apply_dict_mapping() (weblyzard_api.xml_content.XMLContent class method), 5
 as_dict() (weblyzard_api.xml_content.Sentence method), 4
 as_dict() (weblyzard_api.xml_content.XMLContent method), 5
 ATTRIBUTE_MAPPING (weblyzard_api.client.jeremia.Jeremia attribute), 10
 ATTRIBUTE_MAPPING (weblyzard_api.client.jesaja.Jesaja attribute), 11
 ATTRIBUTE_MAPPING (weblyzard_api.client.recognize.Recognize attribute), 19

C

change_log_level() (weblyzard_api.client.jesaja.Jesaja method), 12
 Classifier (class in weblyzard_api.client.classifier), 7
 CLASSIFIER_WS_BASE_PATH (weblyzard_api.client.classifier.Classifier

attribute), 7
 classify() (weblyzard_api.client.classifier.Classifier method), 7
 clear_blacklist() (weblyzard_api.client.jeremia.Jeremia method), 10
 commit() (weblyzard_api.client.jeremia.Jeremia method), 10
 content_id (weblyzard_api.xml_content.XMLContent attribute), 5
 content_type (weblyzard_api.xml_content.XMLContent attribute), 5
 convert() (weblyzard_api.xml_content.XMLContent class method), 5
 convert_document() (weblyzard_api.client.jesaja.Jesaja class method), 12
 convert_document() (weblyzard_api.client.recognize.Recognize class method), 19

D

dependency_list (weblyzard_api.xml_content.Sentence attribute), 4
 DOCS (weblyzard_api.client.recognize.EntityLyizardTest attribute), 17
 DOCS_XML (weblyzard_api.client.recognize.EntityLyizardTest attribute), 18
 DomainSpecificity (class in weblyzard_api.client.domain_specificity), 8

E

EntityLyizardTest (class in weblyzard_api.client.recognize), 17

F

finalize_corpora() (weblyzard_api.client.jesaja.Jesaja method), 13
 finalize_profile() (weblyzard_api.client.jesaja.Jesaja method), 13

G

get_blacklist() (weblyzard_api.client.jeremia.Jeremia

method), 10
get_cache_stats() (weblyzard_api.client.jesaja.Jesaja method), 13
get_cached_corpora() (weblyzard_api.client.jesaja.Jesaja method), 13
get_content_id() (weblyzard_api.xml_content.XMLContent method), 6
get_content_type() (weblyzard_api.xml_content.XMLContent method), 6
get_corpus_size() (weblyzard_api.client.jesaja.Jesaja method), 13
get_dependency_list() (weblyzard_api.xml_content.Sentence method), 4
get_documents() (weblyzard_api.client.jesaja.Jesaja class method), 13
get_domain_specificity() (weblyzard_api.client.domain_specificity.DomainSpecificity method), 8
get_focus() (weblyzard_api.client.recognize.Recognize method), 19
get_keywords() (weblyzard_api.client.jesaja.Jesaja method), 13
get_keywords_xml() (weblyzard_api.client.jesaja.Jesaja method), 13
get_lang() (weblyzard_api.xml_content.XMLContent method), 6
get_nilsimsa() (weblyzard_api.xml_content.XMLContent method), 6
get_plain_text() (weblyzard_api.xml_content.XMLContent method), 6
get_pos_tags() (weblyzard_api.xml_content.Sentence method), 4
get_pos_tags_list() (weblyzard_api.xml_content.Sentence method), 4
get_pos_tags_string() (weblyzard_api.xml_content.Sentence method), 4
get_sentence() (weblyzard_api.xml_content.Sentence method), 4
get_sentences() (weblyzard_api.xml_content.XMLContent method), 6
get_text() (weblyzard_api.xml_content.XMLContent class method), 6
get_title() (weblyzard_api.xml_content.XMLContent method), 6
get_tokens() (weblyzard_api.xml_content.Sentence method), 4
get_xml_doc() (weblyzard_api.client.jeremia.Jeremia method), 10
get_xml_document() (weblyzard_api.client.recognize.Recognize method), 19
get_xml_document() (weblyzard_api.xml_content.XMLContent method), 6
get_xml_version() (weblyzard_api.xml_content.XMLContent class method), 6

H

has_profile() (weblyzard_api.client.domain_specificity.DomainSpecificity method), 8
has_profile() (weblyzard_api.client.jesaja.Jesaja method), 13
hello_world() (weblyzard_api.client.classifier.Classifier method), 7

I

IS_ONLINE (weblyzard_api.client.recognize.EntityLyzardTest attribute), 18

J

Jeremia (class in weblyzard_api.client.jeremia), 9
Jesaja (class in weblyzard_api.client.jesaja), 11

L

label (weblyzard_api.xml_content.LabeledDependency attribute), 3
LabeledDependency (class in weblyzard_api.xml_content), 3
lang (weblyzard_api.xml_content.XMLContent attribute), 6
list_configured_profiles() (weblyzard_api.client.recognize.Recognize method), 19
list_profiles() (weblyzard_api.client.domain_specificity.DomainSpecificity method), 8
list_profiles() (weblyzard_api.client.jesaja.Jesaja method), 13
list_profiles() (weblyzard_api.client.recognize.Recognize method), 19
list_stoplists() (weblyzard_api.client.jesaja.Jesaja method), 13

M

meminfo() (weblyzard_api.client.domain_specificity.DomainSpecificity method), 8
meminfo() (weblyzard_api.client.jesaja.Jesaja method), 13

N

nilsimsa (weblyzard_api.xml_content.XMLContent attribute), 6

O

OUTPUT_FORMATS

blyzard_api.client.recognize.Recognize attribute), 19

P

parent (weblyzard_api.xml_content.LabeledDependency attribute), 3

parse_document() (weblyzard_api.client.sentiment_analysis.SentimentAnalysis method), 21

parse_document_list() (weblyzard_api.client.sentiment_analysis.SentimentAnalysis method), 22

parse_documents() (weblyzard_api.client.domain_specificity.DomainSpecificity method), 9

parse_xml_content() (weblyzard_api.xml_content.XMLContent class method), 6

plain_text (weblyzard_api.xml_content.XMLContent attribute), 6

POS (class in weblyzard_api.client.pos), 14

pos (weblyzard_api.xml_content.LabeledDependency attribute), 3

pos_tag_string (weblyzard_api.xml_content.Sentence attribute), 4

pos_tagging() (weblyzard_api.client.pos.POS method), 14

pos_tags (weblyzard_api.xml_content.Sentence attribute), 4

pos_tags_list (weblyzard_api.xml_content.Sentence attribute), 5

R

Recognize (class in weblyzard_api.client.recognize), 18

remove_profile() (weblyzard_api.client.recognize.Recognize method), 19

reset() (weblyzard_api.client.sentiment_analysis.SentimentAnalysis method), 22

S

search_document() (weblyzard_api.client.recognize.Recognize method), 20

search_documents() (weblyzard_api.client.domain_specificity.DomainSpecificity method), 9

search_documents() (weblyzard_api.client.recognize.Recognize method), 20

search_text() (weblyzard_api.client.recognize.Recognize method), 21

Sentence (class in weblyzard_api.xml_content), 3

sentence (weblyzard_api.xml_content.Sentence attribute), 5

sentences (weblyzard_api.xml_content.XMLContent attribute), 6

SentimentAnalysis (class in weblyzard_api.client.sentiment_analysis), 21

set_dependency_list() (weblyzard_api.xml_content.Sentence method), 5

set_pos_tags() (weblyzard_api.xml_content.Sentence method), 5

set_pos_tags_list() (weblyzard_api.xml_content.Sentence method), 5

set_pos_tags_string() (weblyzard_api.xml_content.Sentence method), 5

set_sentence() (weblyzard_api.xml_content.Sentence method), 5

setUp() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

status() (weblyzard_api.client.jeremia.Jeremia method), 10

status() (weblyzard_api.client.recognize.Recognize method), 21

submit() (weblyzard_api.client.jeremia.Jeremia method), 10

submit_document() (weblyzard_api.client.jeremia.Jeremia method), 10

submit_documents() (weblyzard_api.client.jeremia.Jeremia method), 10

submit_documents_blacklist() (weblyzard_api.client.jeremia.Jeremia method), 10

SUPPORTED_XML_VERSIONS (weblyzard_api.xml_content.XMLContent attribute), 5

test_entity_lyzard() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_focus_search() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_geo() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_missing_profiles() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

T

test_entity_lyzard() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_focus_search() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_geo() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_missing_profiles() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_organization() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_password() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_people() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

test_search_xml() (weblyzard_api.client.recognize.EntityLyzardTest method), 18

TESTED_PROFILES (weblyzard_api.client.recognize.EntityLyzardTest attribute), 18

title (weblyzard_api.xml_content.XMLContent attribute), 6

tokens (weblyzard_api.xml_content.Sentence attribute), 5

train() (weblyzard_api.client.classifier.Classifier method), 7

weblyzard_api.client.domain_specificity (module), 8

weblyzard_api.client.jeremia (module), 9

weblyzard_api.client.jesaja (module), 11

weblyzard_api.client.pos (module), 14

weblyzard_api.client.recognize (module), 17

weblyzard_api.client.sentiment_analysis (module), 21

weblyzard_api.xml_content (module), 3

X

xml (weblyzard_api.client.recognize.EntityLyzardTest attribute), 18

XMLContent (class in weblyzard_api.xml_content), 5

U

update_attributes() (weblyzard_api.xml_content.XMLContent method), 6

update_blacklist() (weblyzard_api.client.jeremia.Jeremia method), 11

update_context() (weblyzard_api.client.sentiment_analysis.SentimentAnalysis method), 22

update_lexicon() (weblyzard_api.client.sentiment_analysis.SentimentAnalysis method), 22

update_negation() (weblyzard_api.client.sentiment_analysis.SentimentAnalysis method), 22

update_sentences() (weblyzard_api.xml_content.XMLContent method), 6

URL_PATH (weblyzard_api.client.domain_specificity.DomainSpecificity attribute), 8

URL_PATH (weblyzard_api.client.jeremia.Jeremia attribute), 10

URL_PATH (weblyzard_api.client.jesaja.Jesaja attribute), 11

URL_PATH (weblyzard_api.client.recognize.Recognize attribute), 19

V

VALID_CORPUS_FORMATS (weblyzard_api.client.jesaja.Jesaja attribute), 11

version() (weblyzard_api.client.jeremia.Jeremia method), 11

W

weblyzard_api (module), 3

weblyzard_api.client (module), 7

weblyzard_api.client.classifier (module), 7