dandelion-eu Documentation

Release 0.3.3

Stefano Parmesan

Aug 07, 2023

Contents

Connect to the dandelion.eu API in a very pythonic way!

Note: This is a beta package, and may change dramatically. Not all the features are implemented as yet, but many are next to come. Please be patient, and perhaps write us to let us know you're out there, waiting for something.

CHAPTER 1

Installation

`dandelion-eu` is available on `https://pypi.python.org/pypi/dandelion-eu/` install it simply with:

```
pip install dandelion-eu
```

CHAPTER 2

Contents

2.1 General Documentation

What is explained here applies to all the services available in this package. For specific documentation on each service, please refer to their page.

2.1.1 Authentication

Most (all?) of the dandelion.eu services require authentication. You can find your authentication token on your dashboard and pass them to the class constructor, for example:

```
>>> from dandelion import DataTXT
>>> administrative_regions = DataTXT(token='8697xxxx8b99xxxxeecbxxxxb163xxxx')
```

If you need to instantiate more services, you can specify your authentication token just once using `dandelion.default_config`:

```
>>> from dandelion import default_config
>>> default_config['token'] = '8697xxxx8b99xxxxeecbxxxxb163xxxx'

>>> from dandelion import DataTXT
>>> datatxt = DataTXT()
```

The old authentication system by *app_id* and *app_key* does not work anymore, but it is still possible to use these parameters by setting the token value on both.

2.1.2 Caching your queries

To avoid wasting units (and time) you can easily cache all your requests using the classes provided by the `dandelion.cache` package. Currently just `FileCache` is available, but more can be easily implemented using the abstract cache class.

To enable caching, instantiate a cache object and pass it to the services you need:

```
>>> from dandelion import DataTXT
>>> from dandelion.cache import FileCache
>>> datatxt = DataTXT(cache=FileCache('.cache_dir'))
```

2.2 The dataTXT API

dataTXT is a family of semantic services developed by SpazioDati. All its methods are available in the same class:

```
>>> from dandelion import DataTXT
>>> datatxt = DataTXT(token='')
```

2.2.1 NEX: Named Entity Extraction

dataTXT-NEX is a named entity extraction & linking API that performs very well even on short texts, on which many other similar services do not. With this API you will be able to automatically tag your texts, extracting Wikipedia entities and enriching your data.

You can extract annotated entities with:

```
>>> for annotation in datatxt.nex('Oh my, arduino is super cool, so #opensource').
↪annotations:
...     print(annotation.uri)
http://en.wikipedia.org/wiki/Arduino
http://en.wikipedia.org/wiki/Open_source
```

Additional parameters can be specified simply by:

```
>>> result = datatxt.nex('Oh my, arduino is super cool, so #opensource',
...                      include_lod=True,
...                      )
>>> [annotation.lod.dbpedia for annotation in result.annotations]
['http://dbpedia.org/resource/Arduino',
 'http://dbpedia.org/resource/Open_source']
```

Check out the dataTXT-NEX documentation on dandelion.eu for more information about what can be done with NEX.

2.2.2 SIM: Text Similarity

dataTXT-SIM is a semantic sentence similarity API optimized on short sentences. With this API you will be able to compare two sentences and get a score of their semantic similarity. It works even if the two sentences don't have any word in common.

You can compute the semantic similarity between two texts with:

```
>>> datatxt.sim('Barack Obama is the president of the US',
...             'Bob Iger is the CEO of Walt Disney')
{'lang': 'en',
 'langConfidence': 1.0,
 'similarity': 0.2564,
 'time': 11,
 'timestamp': '2042-01-01T01:02:03'}
```

Check out the dataTXT-SIM documentation on dandelion.eu for more information about what can be done with SIM.

2.2.3 LI: Language Identification

dataTXT-LI is a simple language identification API; it is a tool that may be useful when dealing with texts, so we decided to open it to all our users. It currently supports more than 50 languages.

You can identify the language of a text with:

```
>>> datatxt.li('mamma mia! un testo in italiano!')
{'detectedLangs': [{'confidence': 0.9999952605110598, 'lang': 'it'}],
 'time': 0,
 'timestamp': '2042-01-01T01:02:03'}
```

Check out the dataTXT-LI documentation on dandelion.eu.