Django Dandelion Documentation

Release 0.1.4

Alessio Bazzanella

Jun 29, 2017

Contents

1 Installation **3**

2 Contents **5**

2.1 Django Dandelion 5
2.2 Quick Start 6
2.3 Contributing 11
2.4 Credits 13
2.5 Change Log 13

Use the Dandelion API with Django

CHAPTER 1

Installation

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

```
$ pip install django-dandelion
```

Add the entry DANDELION_TOKEN. The recommended method is to setup your production keys using environment variables. This helps to keep them more secure. Your test keys can be displayed in your code directly.

The following entry look for your DANDELION_TOKEN in your environment and, if it can't find them,uses your test keys values instead:

```
DANDELION_TOKEN = os.environ.get("DANDELION_TOKEN", "<your dandelion token>")
```

CHAPTER 2

Contents

Django Dandelion

Use the Dandelion API with Django

Documentation

The full documentation is at https://django-dandelion.readthedocs.io.

Quickstart

Install Django Dandelion:

```
$ pip install django-dandelion
```

Add `django_dandelion` to your `INSTALLED_APPS`

```
INSTALLED_APPS = [
    # ...
    'django_dandelion',
]
```

Add the entry DANDELION_TOKEN. The recommended method is to setup your production keys using environment variables. This helps to keep them more secure. Your test keys can be displayed in your code directly.

The following entry look for your DANDELION_TOKEN in your environment and, if it can't find them,uses your test keys values instead:

```
DANDELION_TOKEN = os.environ.get("DANDELION_TOKEN", "<your dandelion token>")
```

Register on dandelion to obtain your authentication keys and enrich your application with our semantic intelligence.

You can also change the url of the host and decide whether to use the cache:

```
DANDELION_HOST = 'api.dandelion.eu'  # Default 'api.dandelion.eu'
DANDELION_USE_CACHE = True  # Default True
```

Running Tests

Does the code actually work?

```
# Add "export DANDELION_TOKEN=<your dandelion token>" to <YOURVIRTUALENV>/bin/activate
$ source <YOURVIRTUALENV>/bin/activate
$ (myenv) $ pip install tox
$ (myenv) $ tox
```

Quick Start

Install

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

```
$ pip install django-dandelion
```

Configure

Settings

Add `django_dandelion` to your `INSTALLED_APPS`

```
INSTALLED_APPS = [
    # ...
    'django_dandelion',
]
```

Add the entry DANDELION_TOKEN. The recommended method is to setup your production keys using environment variables. This helps to keep them more secure. Your test keys can be displayed in your code directly.

The following entry look for your DANDELION_TOKEN in your environment and, if it can't find them,uses your test keys values instead:

```
DANDELION_TOKEN = os.environ.get("DANDELION_TOKEN", "<your dandelion token>")
```

Register on dandelion to obtain your authentication keys and enrich your application with our semantic intelligence.

You can also change the url of the host and decide whether to use the cache:

```
DANDELION_HOST = 'api.dandelion.eu'  # Default 'api.dandelion.eu'
DANDELION_USE_CACHE = True  # Default True
```

Requests

DataTXT

- EntityExtraction (Documentation)

```
>>> from django_dandelion.datatxt import EntityExtraction
>>> entityExtraction = EntityExtraction()
>>> entityExtraction.params = 'text', u'They say Apple is better than Windows'
>>> entityExtraction.params = 'lang', u'en'
>>> entityExtraction.analyze()
{u'annotations': [{u'confidence': 0.7353,
   u'end': 14,
   u'id': 856,
   u'label': u'Apple',
   u'spot': u'Apple',
   u'start': 9,
   u'title': u'Apple Inc.',
   u'uri': u'http://en.wikipedia.org/wiki/Apple_Inc.'},
  {u'confidence': 0.6904,
   u'end': 37,
   u'id': 18890,
   u'label': u'Microsoft Windows',
   u'spot': u'Windows',
   u'start': 30,
   u'title': u'Microsoft Windows',
   u'uri': u'http://en.wikipedia.org/wiki/Microsoft_Windows'}],
 u'lang': u'en',
 u'time': 0,
 u'timestamp': u'2017-03-09T16:10:46.703'}

>>> entityExtraction = EntityExtraction()
>>> spots = entityExtraction.UserDefinedSpots()  # spots = EntityExtraction.
↪UserDefinedSpots()
>>> obj = spots.create(data=u'{"description": "My botanical custom spots","lang": "en
↪","list": [{"spot": "làres","topic": "Larix decidua"},{"spot": "stropèr","topic":
↪"Salix viminalis"},{"spot": "noselèr","topic": "Corylus avellana"},{"spot": "pomèr",
↪"topic": "Malus domestica"},{"spot": "brugnèra","topic": "Prunus domestica"}]}')
>>> obj
{u'created': u'2017-03-10T11:34:45',
 u'data': {u'defaults': {u'exactMatch': False,
   u'greedy': True,
   u'namedEntity': False},
  u'description': u'My botanical custom spots',
  u'lang': u'en',
  u'list': [{u'exactMatch': False,
    u'greedy': True,
    u'namedEntity': False,
    u'spot': u'l\xe0res',
    u'topic': u'Larix decidua'},
   {u'exactMatch': False,
    u'greedy': True,
    u'namedEntity': False,
    u'spot': u'strop\xe8r',
    u'topic': u'Salix viminalis'},
   {u'exactMatch': False,
    u'greedy': True,
    u'namedEntity': False,
```

```
    u'spot': u'nosel\xe8r',
    u'topic': u'Corylus avellana'},
   {u'exactMatch': False,
    u'greedy': True,
    u'namedEntity': False,
    u'spot': u'pom\xe8r',
    u'topic': u'Malus domestica'},
   {u'exactMatch': False,
    u'greedy': True,
    u'namedEntity': False,
    u'spot': u'brugn\xe8ra',
    u'topic': u'Prunus domestica'}]},
 u'dataType': u'custom-spots',
 u'id': u'b1290d8a-4af3-4c70-85cc-1eb6a97f7725',
 u'modified': u'2017-03-10T11:34:45',
 u'timestamp': u'2017-03-09T16:10:46.703'}
>>> entityExtraction.params = 'text', u'Larix decidua, commonly called European or␣
↪common larch, is a deciduous conifer although it looks like a needled evergreen in␣
↪summer. It is a large tree that will grow to 60-100’ tall with a pyramidal shape,␣
↪horizontal branching and drooping branchlets.'
>>> entityExtraction.params = 'custom_spots', obj.id
>>> entityExtraction.params = 'min_confidence', 0.8
>>> entityExtraction.analyze()
{u'annotations': [{u'confidence': 0.8759,
   u'end': 13,
   u'id': 1584580,
   u'label': u'European larch',
   u'spot': u'Larix decidua',
   u'start': 0,
   u'title': u'Larix decidua',
   u'uri': u'http://en.wikipedia.org/wiki/Larix_decidua'},
  {u'confidence': 0.8184,
   u'end': 55,
   u'id': 99384,
   u'label': u'Larch',
   u'spot': u'larch',
   u'start': 50,
   u'title': u'Larch',
   u'uri': u'http://en.wikipedia.org/wiki/Larch'},
  {u'confidence': 0.829,
   u'end': 71,
   u'id': 66722,
   u'label': u'Deciduous',
   u'spot': u'deciduous',
   u'start': 62,
   u'title': u'Deciduous',
   u'uri': u'http://en.wikipedia.org/wiki/Deciduous'},
  {u'confidence': 0.9061,
   u'end': 79,
   u'id': 68085,
   u'label': u'Conifer',
   u'spot': u'conifer',
   u'start': 72,
   u'title': u'Pinophyta',
   u'uri': u'http://en.wikipedia.org/wiki/Pinophyta'}],
 u'lang': u'en',
 u'langConfidence': 1.0,
 u'time': 12,
```

```
 u'timestamp': u'2017-03-09T16:10:46.703'}
```

- TextSimilarity (Documentation)

```
>>> from django_dandelion.datatxt import TextSimilarity
>>> textSimilarity = TextSimilarity()
>>> textSimilarity.params = 'text1', u'Reports that the NSA eavesdropped on world␣
↪leaders have "severely shaken" relations between Europe and the U.S., German␣
↪Chancellor Angela Merkel said.'
>>> textSimilarity.params = 'text2', u'Germany and France are to seek talks with the␣
↪US to settle a row over spying, as espionage claims continue to overshadow an EU␣
↪summit in Brussels.'
>>> textSimilarity.analyze()
{u'annotations': [{u'confidence': 0.7353,
   u'end': 14,
   u'id': 856,
   u'label': u'Apple',
   u'spot': u'Apple',
   u'start': 9,
   u'title': u'Apple Inc.',
   u'uri': u'http://en.wikipedia.org/wiki/Apple_Inc.'},
  {u'confidence': 0.6904,
   u'end': 37,
   u'id': 18890,
   u'label': u'Microsoft Windows',
   u'spot': u'Windows',
   u'start': 30,
   u'title': u'Microsoft Windows',
   u'uri': u'http://en.wikipedia.org/wiki/Microsoft_Windows'}],
 u'lang': u'en',
 u'time': 0,
 u'timestamp': u'2017-03-09T16:10:46.703'}
```

- TextClassification (Documentation)

```
>>> from django_dandelion.datatxt import TextClassification
>>> textClassification = TextClassification()
>>> classifier = textClassification.UserDefinedClassifiers()  # classifier =␣
↪TextClassification.UserDefinedClassifiers()
>>> obj = classifier.create(data=u'{"description": "My first model for classifying␣
↪news","lang": "en","categories": [{"name": "Sport","topics": {"http://en.wikipedia.
↪org/wiki/Sport": 2.0,"http://en.wikipedia.org/wiki/Baseball": 1.0,"http://en.
↪wikipedia.org/wiki/Basketball": 1.0,"http://en.wikipedia.org/wiki/Football": 1.0}},{
↪"name": "Politics","topics": {"Politics": 2.0,"Politician": 1.5,"David Cameron": 1.
↪0,"Angela Merkel": 1.0}}]}')
>>> obj
{u'created': u'2017-03-09T16:10:46.703',
 u'data': {u'categories': [{u'name': u'Sport',
    u'topics': {u'http://en.wikipedia.org/wiki/Baseball': 1.0,
     u'http://en.wikipedia.org/wiki/Basketball': 1.0,
     u'http://en.wikipedia.org/wiki/Football': 1.0,
     u'http://en.wikipedia.org/wiki/Sport': 2.0}},
   {u'name': u'Politics',
    u'topics': {u'Angela Merkel': 1.0,
     u'David Cameron': 1.0,
     u'Politician': 1.5,
     u'Politics': 2.0}}],
  u'description': u'My first model for classifying news',
```

```
  u'lang': u'en'},
 u'dataType': u'cl-model',
 u'id': u'7a5a4c4f-8e4a-484a-9f65-b3240819bcfa',
 u'modified': u'2017-03-09T16:36:02',
 u'timestamp': u'2017-03-09T16:10:46.703'}
>>> textClassification.params = 'text', u'See how the main parties are doing in the␣
↪latest opinion polls on voting intention'
>>> textClassification.params = 'model', obj.id
>>> textClassification.analyze()
{u'categories': [{u'name': u'Politics', u'score': 0.34198442},
  {u'name': u'Sport', u'score': 0.10255624}],
 u'lang': u'en',
 u'time': 1,
 u'timestamp': u'2017-03-09T16:10:46.703'}
```

- LanguageDetection (Documentation)

```
>>> from django_dandelion.datatxt import LanguageDetection
>>> languageDetection = LanguageDetection()
>>> languageDetection.params = 'text', u'Le nostre tre M sono: mafia, mamma, mandolino
↪'
>>> languageDetection.analyze()
{u'detectedLangs': [{u'confidence': 1.0, u'lang': u'it'}],
 u'time': 1,
 u'timestamp': u'2017-03-09T16:10:46.703'}
```

- SentimentAnalysis (Documentation)

```
>>> from django_dandelion.datatxt import SentimentAnalysis
>>> sentimentAnalysis = SentimentAnalysis()
>>> sentimentAnalysis.params = 'text', u'Grande applicazione, grande social! Viva␣
↪Twitter'
>>> sentimentAnalysis.analyze()
{u'lang': u'it',
 u'langConfidence': 1.0,
 u'sentiment': {u'score': 0.85, u'type': u'positive'},
 u'time': 0,
 u'timestamp': u'2017-03-09T16:10:46.703'}
```

Datagraph

- Wikisearch (Documentation)

```
>>> from django_dandelion.datagraph import Wikisearch
>>> wikisearch = Wikisearch()
>>> wikisearch.params = 'text', u'Duomo di Trento'
>>> wikisearch.params = 'lang', u'it'
>>> wikisearch.params = 'limit', 3
>>> wikisearch.analyze()
{u'count': 1117,
 u'entities': [{u'id': 925191,
   u'label': u'Cattedrale di San Vigilio',
   u'title': u'Cattedrale di San Vigilio',
   u'uri': u'http://it.wikipedia.org/wiki/Cattedrale_di_San_Vigilio',
   u'weight': 35.093304},
  {u'id': 730261,
```

```
   u'label': u'Duomo',
   u'title': u'Duomo',
   u'uri': u'http://it.wikipedia.org/wiki/Duomo',
   u'weight': 30.126003},
  {u'id': 100137,
   u'label': u'Trento',
   u'title': u'Trento',
   u'uri': u'http://it.wikipedia.org/wiki/Trento',
   u'weight': 23.563704}],
 u'lang': u'it',
 u'query': u'full',
 u'time': 3,
 u'timestamp': u'2017-03-09T16:10:46.703'}
```

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/AlessioBazzanella/django-dandelion/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

Write Documentation

Django Dandelion could always use more documentation, whether as part of the official Django Dandelion docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/AlessioBazzanella/django-dandelion/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *django-dandelion* for local development.

1. Fork the *django-dandelion* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-dandelion.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-dandelion
$ cd django-dandelion/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 django_dandelion tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/AlessioBazzanella/django-dandelion/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_django_dandelion
```

Credits

Development Lead

- Alessio Bazzanella <alessio.bazzanella@me.com>

Contributors

None yet. Why not be the first?

Change Log

All notable changes to this project will be documented in this file.

The format is based on KeepAChangelog and this project adheres to SemanticVersioning.

[0.1.4] - 2017-06-29

Added

- Added `top_entities` and `epsilon` params to `EntityExtraction`
- Added `nex.top_entities` and `nex.epsilon` params to `TextSimilarity`
- Added `nex.top_entities`, `nex.min_confidence`, `nex.min_length`, `nex.social.hashtag`, `nex.social.mention`, `nex.include`,

`nex.extra_types`, `nex.country`, `nex.custom_spots` and `nex.epsilon` params to `TextSimilarity`

[0.1.3] - 2017-03-14

Fixed

- Merge extra_dict with params in BaseDandelionParamsRequest._do_request

- Fixed app name in documentation to add to `INSTALLED_APPS`

[0.1.2] - 2017-03-10

Added

- User-defined spots (EntityExtraction.UserDefinedSpots())

[0.1.1] - 2017-03-10

Fixed

- PyPI version release

[0.1.0] - 2017-03-10

- First release on PyPI.