# DateParser Documentation

***Release 0.1.0***

**Scrapinghub**

June 17, 2015

Contents

Contents:

CHAPTER 1

# DateParser

Date parsing library designed to make it easy parsing dates commonly found in web pages

- Free software: BSD license
- Documentation: https://dateparser.readthedocs.org.

## 1.1 Features

If you have needed to parse dates before, you probably have used the date parser of the dateutil module. We built this library on top of it, adding a few features:

- dateparser support dates in languages other than English
- in fact, it can detect the language automatically
- it can give you the date for text like: `'1 min ago'`,`'2 weeks ago'`,`'3 months, 1 weeks and 1 day ago'`, etc

The goal is to support the common date formats used in websites all around the world.

## 1.2 Limitations

DateParser currently tries hard to get the date information right (year, month and day), but it has limited support for parsing time (hours, minutes and seconds).

CHAPTER 2

# Installation

At the command line:

```
$ easy_install dateparser
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv dateparser
$ pip install dateparser
```

CHAPTER 3

# Usage

The most common way is to use the `dateparser.date.DateDataParser` class, that wraps around most of the functionality in the module.

Here is a quick example of usage:

```
>>> from dateparser.date import DateDataParser
>>> ddp = DateDataParser()
>>> ddp.get_date_data('1 min ago')
{'date_obj': datetime.datetime(2014, 8, 20, 21, 1, 42, 590596), 'period': u'day'}
>>> ddp.get_date_data('1 week ago')
{'date_obj': datetime.datetime(2014, 8, 13, 21, 2, 42, 590596), 'period': u'weeks'}
>>> ddp.get_date_data('1 year ago')
{'date_obj': datetime.datetime(2013, 8, 20, 21, 2, 42, 590596), 'period': u'years'}
>>> ddp.get_date_data('12/12/12')
{'date_obj': datetime.datetime(2012, 12, 12, 0, 0), 'period': 'day'}
>>> ddp.get_date_data('13 August, 2014')
{'date_obj': datetime.datetime(2014, 8, 13, 0, 0), 'period': 'day'}
```

## 3.1 Note about language detection

As it is now, an instance of DateDataParser by default assumes that all of the dates fed to it will be in the same language.

So, it will keep trying to detect the possible languages until it reduces it to only one possibility. When it does, it will just assume that for the next dates and won't try to execute the language detection again:

```
>>> ddp = DateDataParser()
>>> ddp.get_date_data('1 minuto atrás')
{'date_obj': datetime.datetime(2014, 8, 20, 21, 1, 42, 590596), 'period': u'day'}
>>> ddp.get_date_data('13 Agosto 2014')
{'date_obj': datetime.datetime(2014, 8, 13, 0, 0), 'period': 'day'}
>>> ddp.get_date_data('13 Marzo 2014')
{'date_obj': datetime.datetime(2014, 3, 13, 0, 0), 'period': 'day'}
>>> ddp.get_date_data('13 Maio 2014')
Traceback (most recent call last):
...
dateparser.date_parser.LanguageWasNotSeenBeforeError
```

If you want it to redetect the language every time, you can use a custom date_parser, like so:

```
>>> ddp = DateDataParser()
>>> from dateparser.date_parser import DateParser
>>> ddp.date_parser = DateParser(allow_redetect_language=True)
>>> ddp.get_date_data('1 minuto atrás')
{'date_obj': datetime.datetime(2014, 8, 20, 21, 1, 42, 590596), 'period': u'day'}
>>> ddp.get_date_data('13 Agosto 2014')
{'date_obj': datetime.datetime(2014, 8, 13, 0, 0), 'period': 'day'}
>>> ddp.get_date_data('13 Marzo 2014')
{'date_obj': datetime.datetime(2014, 3, 13, 0, 0), 'period': 'day'}
>>> ddp.get_date_data('13 Maio 2014')
{'date_obj': datetime.datetime(2014, 5, 13, 0, 0), 'period': 'day'}
```

## 3.2 How to use in a Scrapy Cloud project

To use in Scrapy Cloud, first you need to build an egg for the library.

Clone the repo and inside its directory, run the command:

```
python setup.py bdist_egg
```

After that, you can upload the egg using Scrapy Cloud's Dashboard interface, or you can use shubc command and do:

```
shubc eggs-add <YOUR_PROJECT_ID> dist/dateparser-0.1-py2.7.egg
```

CHAPTER 4

# 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:

## 4.1 Types of Contributions

### 4.1.1 Report Bugs

Report bugs at https://github.com/scrapinghub/dateparser/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.

### 4.1.2 Fix Bugs

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

### 4.1.3 Implement Features

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

### 4.1.4 Write Documentation

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

### 4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/scrapinghub/dateparser/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 contributions are welcome :)

## 4.2 Get Started!

Ready to contribute? Here's how to set up *dateparser* for local development.

1. Fork the *dateparser* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dateparser.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 dateparser
$ cd dateparser/
$ 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:

```
$ pip install -r tests/requirements.txt # install test dependencies
$ flake8 dateparser tests
$ nosetests
$ 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.

## 4.3 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. Check https://travis-ci.org/scrapinghub/dateparser/pull_requests and make sure that the tests pass for all supported Python versions.

## 4.4 Tips

To run a subset of tests:

```
$ nosetests tests.test_date_parser
```

CHAPTER 5

# Credits

## 5.1 Development Lead

- Claudio Salazar <claudio@scrapinghub.com>
- Elias Dorneles <elias@scrapinghub.com>
- Eugene Amirov <eugene@scrapinghub.com>

## 5.2 Contributors

Shuai Lin, Ismael Carnales and other folks from Scrapinghub staff.

CHAPTER 6

# History

CHAPTER 7

## 0.1.0 (2014-11-24)

- First release on PyPI.

CHAPTER 8

# Indices and tables

- genindex
- modindex
- search