FlexMatcher Documentation

Release 0.8.0

BigGorilla Team

Jan 11, 2018

Contents

FlexMatcher is a schema matching package in Python which handles the problem of matching multiple schemas to a single mediated schema. FlexMatcher is part of the *BigGorilla <http://biggorilla.org>* project which provides tools for data integration and data preparation.

- Free software: Apache Software License 2.0

Contents:

CHAPTER 1

Installation

1.1 Stable release

To install FlexMatcher, run this command in your terminal:

```
$ pip install flexmatcher
```

This is the preferred method to install FlexMatcher, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for FlexMatcher can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/biggorilla-gh/flexmatcher
```

Or download the tarball:

```
$ curl  -OL https://github.com/biggorilla-gh/flexmatcher/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 2

Examples

Imagine that we have two datasets on movies provided in pandas dataframe format. Let's say that we are interested in three attributes, namely 'movie_name', 'movie_year' and 'movie_rating'. The name of columns in the two datasets may differ form the names that we just listed. Let's say that we look into the first two dataset and specify how each column maps to one of the three attributes that is of interest to us.:

```
vals1 = [['year', 'Movie', 'imdb_rating'],
        ['2001', 'Lord of the Rings', '8.8'],
        ['2010', 'Inception', '8.7'],
        ['1999', 'The Matrix', '8.7']]
header = vals1.pop(0)
data1 = pd.DataFrame(vals1, columns=header)
# creating the second dataset
vals2 = [['title', 'produced', 'popularity'],
        ['The Godfather', '1972', '9.2'],
        ['Silver Linings Playbook', '2012', '7.8'],
        ['The Big Short', '2015', '7.8']]
header = vals2.pop(0)
data2 = pd.DataFrame(vals2, columns=header)
# specifying the mappings for the first and second datasets
data1_mapping = {'year': 'movie_year',
                 'imdb_rating': 'movie_rating',
                 'Movie': 'movie_name'}
data2_mapping = {'popularity': 'movie_rating',
                 'produced': 'movie_year',
                 'title': 'movie_name'}
```

Now, let's assume that we are given a thirs dataset.:

```
# creating the third dataset
vals3 = [['rt', 'id', 'yr'],
        ['8.5', 'The Pianist', '2002'],
        ['7.7', 'The Social Network', '2010']]
header = vals3.pop(0)
data3 = pd.DataFrame(vals3, columns=header)
```

We can use flexmatcher to find how the columns in the new dataset are related to the attributes of our interest. To do so, we need to create an instance of FlexMatcher, make a list of available datasets and their mappings to the desired attributes, and train the FlexMatcher.:

```
schema_list = [data1, data2]
mapping_list = [data1_mapping, data2_mapping]
fm = flexmatcher.FlexMatcher(schema_list, mapping_list, sample_size=100)
fm.train()
```

Then, we can use the trained FlexMatcher to predict the mappings for the third dataset as follows.:

```
predicted_mapping = fm.make_prediction(data3)
```

The result is a dictionary that maps every column to an attribute. For instance,:

```
>>> print(predicted_mapping['rc'])
movie_rating
```

CHAPTER 3

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:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/biggorilla-gh/flexmatcher/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.

3.1.2 Fix Bugs

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

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

3.1.4 Write Documentation

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

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/biggorilla-gh/flexmatcher/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 :)

3.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/flexmatcher.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 flexmatcher
$ cd flexmatcher/
$ 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 flexmatcher tests
$ python setup.py test or 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.

3.3 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. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/biggorilla-gh/flexmatcher/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

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

CHAPTER 4

Indices and tables

- genindex
- modindex