fuzzysearch Documentation

Release 0.3.0

Tal Einat

June 30, 2015

Contents

Contents:

CHAPTER 1

fuzzysearch

fuzzysearch is useful for finding approximate subsequence matches

- Free software: MIT license
- Documentation: http://fuzzysearch.rtfd.org.

1.1 Installation

Just install using pip:

```
$ pip install fuzzysearch
```

1.2 Features

- Fuzzy sub-sequence search: Find parts of a sequence which match a given sub-sequence up to a given maximum Levenshtein distance.
- Set individual limits for the number of substitutions, insertions and/or deletions allowed for a near-match.
- Includes optimized implementations for specific use-cases, e.g. only allowing substitutions in near-matches.

1.3 Simple Example

You can usually just use the *find_near_matches()* utility function, which chooses a suitable fuzzy search implementation according to the given parameters:

```
>>> from fuzzysearch import find_near_matches
>>> find_near_matches('PATTERN', 'aaaPATERNaaa', max_l_dist=1)
[Match(start=3, end=9, dist=1)]
```

1.4 Advanced Example

If needed you can choose a specific search implementation, such as *find_near_matches_with_ngrams()*:

```
>>> sequence = '''\
GACTAGCACTGTAGGGATAACAATTTCACACAGGTGGACAATTACATTGAAAATCACAGATTGGTCACACACACA
TTGGACATACATAGAAACACACACACATACATTAGATACGAACATAGAAACACACATTAGACGCGTACATAGACA
CAAACACATTGACAGGCAGTTCAGATGATGACGCCCGACTGATACTCGCGTAGTCGTGGGAGGCAAGGCACACAG
GGGATAGG'''
>>> subsequence = 'TGCACTGTAGGGATAACAAT' #distance 1
>>> max_distance = 2

>>> from fuzzysearch import find_near_matches_with_ngrams
>>> find_near_matches_with_ngrams(subsequence, sequence, max_distance)
[Match(start=3, end=24, dist=1)]
```

CHAPTER 2

Installation

At the command line:

```
$ pip install fuzzysearch
```

Installation should succeed even if building the C extensions fails. If not, you can force the installation to skip building the extensions:

```
$ pip install fuzzysearch --noexts
```

CHAPTER 3

Usage

3.1 Simple Example

You can usually just use the *find_near_matches()* utility function, which chooses a suitable fuzzy search implementation according to the given parameters:

```
>>> from fuzzysearch import find_near_matches
>>> find_near_matches('PATTERN', 'aaaPATERNaaa', max_l_dist=1)
[Match(start=3, end=9, dist=1)]
```

3.2 Advanced Example

If needed you can choose a specific search implementation, such as *find_near_matches_with_ngrams()*:

```
>>> sequence = '''\
GACTAGCACTGTAGGGATAACAATTTCACACAGGTGGACAATTACATTGAAAATCACAGATTGGTCACACACACA
TTGGACATACATAGAAACACACACACATACATTAGATACGAACATAGAAACACACATTAGACGCGTACATAGACA
CAAACACATTGACAGGCAGTTCAGATGATGACGCCCGACTGATACTCGCGTAGTCGTGGGAGGCAAGGCACACAG
GGGATAGG'''
>>> subsequence = 'TGCACTGTAGGGATAACAAT' #distance 1
>>> max_distance = 2

>>> from fuzzysearch import find_near_matches_with_ngrams
>>> find_near_matches_with_ngrams(subsequence, sequence, max_distance)
[Match(start=3, end=24, dist=1)]
```

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/taleinat/fuzzysearch/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

fuzzysearch could always use more documentation, whether as part of the official fuzzysearch 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/taleinat/fuzzysearch/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 :)

4.2 Get Started!

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

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

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

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

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Tal Einat <taleinat@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.3.0 (2015-02-12)

- Added C extensions for several search functions as well as internal functions
- Use C extensions if available, or pure-Python implementations otherwise
- setup.py attempts to build C extensions, but installs without if build fails
- Added `--noexts` setup.py option to avoid trying to build the C extensions
- Greatly improved testing and coverage

6.2 0.2.2 (2014-03-27)

- Added support for searching through BioPython Seq objects
- Added specialized search function allowing only subsitutions and insertions
- Fixed several bugs

6.3 0.2.1 (2014-03-14)

- Fixed major match grouping bug

6.4 0.2.0 (2013-03-13)

- New utility function `find_near_matches()` for easier use
- Additional documentation

6.5 0.1.0 (2013-11-12)

- Two working implementations
- Extensive test suite; all tests passing

- Full support for Python 2.6-2.7 and 3.1-3.3
- Bumped status from Pre-Alpha to Alpha

6.6 0.0.1 (2013-11-01)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search