hashedindex Documentation

Release 0.4.4

Michael Aquilina

Apr 08, 2018

Contents

Contents:

CHAPTER 1

hashedindex

Fast and simple InvertedIndex implementation using hash lists (python dictionaries).

Supports Python 2.7 and Python 3.3+

Free software: BSD license

1.1 Installing

The easiest way to install hashindex is through pypi

```
pip install hashedindex
```

1.2 Features

`hashedindex` provides a simple to use inverted index structure that is flexible enough to work with all kinds of use cases.

Basic Usage:

```
import hashedindex
index = hashedindex.HashedIndex()

index.add_term_occurrence('hello', 'document1.txt')
index.add_term_occurrence('world', 'document1.txt')

index.get_documents('hello')
Counter({'document1.txt': 1})

index.items()
{'hello': Counter({'document1.txt': 1}),
'world': Counter({'document1.txt': 1})}

example = 'The Quick Brown Fox Jumps Over The Lazy Dog'

for term in example.split():
    index.add_term_occurrence(term, 'document2.txt')
```

hashedindex is not limited to strings, any hashable object can be indexed.

```
index.add_term_occurrence('foo', 10)
index.add_term_occurrence(('fire', 'fox'), 90.2)

index.items()
{'foo': Counter({10: 1}), ('fire', 'fox'): Counter({90.2: 1})}
```

1.3 Text Parsing

The hashedindex module comes included with a powerful textparser module with methods to split text into tokens.

```
from hashedindex import textparser
list(textparser.word_tokenize("hello cruel world"))
[('hello',), ('cruel',), ('world',)]
```

Tokens are wrapped within tuples due to the ability to specify any number of n-grams required:

```
list(textparser.word_tokenize("Life is about making an impact, not making an income.",
↪ ngrams=2))
[(u'life', u'is'), (u'is', u'about'), (u'about', u'making'), (u'making', u'an'), (u'an
↪', u'impact'),
 (u'impact', u'not'), (u'not', u'making'), (u'making', u'an'), (u'an', u'income')]
```

Take a look at the function's docstring for information on how to use stopwords, specify a min_length or ignore_numeric terms.

1.4 Integration with Numpy and Pandas

The idea behind hashedindex is to provide a really quick and easy way of generating matrices for machine learning with the additional use of numpy, pandas and scikit-learn. For example:

```
from hashedindex import textparser
import hashedindex
import numpy as np
```

```
index = hashedindex.HashedIndex()

documents = ['spam1.txt', 'ham1.txt', 'spam2.txt']
for doc in documents:
    with open(doc, 'r') as fp:
         for term in textparser.word_tokenize(fp.read()):
             index.add_term_occurrence(term, doc)

# You *probably* want to use scipy.sparse.csr_matrix for better performance
X = np.as_matrix(index.generate_feature_matrix(mode='tfidf'))

y = []
for doc in index.documents():
    y.append(1 if 'spam' in doc else 0)
y = np.asarray(doc)

from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(X, y)
```

You can also extend your feature matrix to a more verbose pandas DataFrame:

```
import pandas as pd
X  = index.generate_feature_matrix(mode='tfidf')
df = pd.DataFrame(X, columns=index.terms(), index=index.documents())
```

All methods within the code have high test coverage so you can be sure everything works as expected.

1.5 Reporting Bugs

Found a bug? Nice, a bug found is a bug fixed. Open an Issue or better yet, open a pull request.

CHAPTER 2

Installation

At the command line:

```
$ easy_install hashedindex
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv hashedindex
$ pip install hashedindex
```

CHAPTER 3

Usage

To use hashedindex in a project:

```
import hashedindex
```

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/MichaelAquilina/hashedindex/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

hashedindex could always use more documentation, whether as part of the official hashedindex 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/MichaelAquilina/hashedindex/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 *hashedindex* for local development.

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

```
$ git clone git@github.com:your_name_here/hashedindex.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 hashedindex
$ cd hashedindex/
$ 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 hashedindex 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, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/MichaelAquilina/hashedindex/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_hashedindex
```

CHAPTER 5

Credits

5.1 Development Lead

- Michael Aquilina <michaelaquilina@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

0.1.0 (2015-01-11)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search