gitcoach Documentation

Release 0.2.0

Mike Hoye

November 09, 2013

Contents

Contents:

CHAPTER 1

gitcoach

Gitlearn and gitcoach are a pair of tools for helping me, and hopefully other people, better understand large projects living in Git by trying to identify codependent pieces of code.

- Free software: BSD license
- Documentation: http://gitcoach.rtfd.org.

1.1 Installation

The easiest way to install gitcoach is through pip:

```
$ pip install gitcoach
```

1.2 Usage

To generate the prediction data, run *gitlearn*. This might take a long time:

```
usage: gitlearn [-h]

Generate coaching data for gitcoach.

optional arguments:
  -h, --help  show this help message and exit
```

The *gitcoach* utility:

```
usage: gitcoach [-h] [--file FILE] [--commit COMMIT] [--threshold THRESHOLD]

Find co-dependent files based on git history. Two files are co-dependent if
they have been modified in the same commits often enough.

optional arguments:
-h, --help            show this help message and exit
--file FILE, -f FILE  Find suggestions for a specific file
--commit COMMIT, -c COMMIT
                      Find suggestions for files modified in a specific
```

```
                        commit.
--threshold THRESHOLD, -t THRESHOLD
                        Threshold for co-incidence ratio (default=0.8).
```

Example output of *gitcoach*:

```
Here are some files you might want to look at:

travis.yml  suggested by    README.rst (0.500000)
tox.ini     suggested by    README.rst (0.500000)
setup.py    suggested by    README.rst (0.500000)
requirements.txt    suggested by    README.rst (0.500000)
```

CHAPTER 2

Installation

At the command line:

```
$ easy_install gitcoach
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv gitcoach
$ pip install gitcoach
```

CHAPTER 3

Usage

To use gitcoach in a project:

```
import gitcoach
```

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/mhoye/gitcoach/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

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

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

   ```
   $ git clone git@github.com:your_name_here/gitcoach.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 gitcoach
   $ cd gitcoach/
   $ 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 gitcoach tests
   $ python setup.py 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. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/mhoye/gitcoach/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 5

Credits

5.1 Development Lead

- Mike Hoye <mhoye@mozilla.com>

5.2 Contributors

- Tavish Armstrong <tavisharmstrong@gmail.com>

CHAPTER 6

History

TODO

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*