BinaryOrNot Documentation

Release 0.4.4

Audrey Roy

Aug 08, 2017

Contents

Contents:

CHAPTER 1

BinaryOrNot

Ultra-lightweight pure Python package to guess whether a file is binary or text, using a heuristic similar to Perl's *pp_fttext* and its analysis by @eliben.

- Free software: BSD license
- Documentation: https://binaryornot.readthedocs.io

Status

It works, and people are using this package in various places. But it doesn't cover all edge cases yet.

The code could be improved. Pull requests welcome! As of now, it is based on these snippets, but that may change:

- http://stackoverflow.com/questions/898669/how-can-i-detect-if-a-file-is-binary-non-text-in-python
- http://stackoverflow.com/questions/1446549/how-to-identify-binary-and-text-files-using-python
- http://code.activestate.com/recipes/173220/
- http://eli.thegreenplace.net/2011/10/19/perls-guess-if-file-is-text-or-binary-implemented-in-python/

Features

Has tests for these file types:

- Text: .txt, .css, .json, .svg, .js, .lua, .pl, .rst
- Binary: .png, .gif, .jpg, .tiff, .bmp, .DS_Store, .eot, .otf, .ttf, .woff, .rgb

Has tests for numerous encodings.

Why?

You may be thinking, "I can write this in 2 lines of code?!"

It's actually not that easy. Here's a great article about how Perl's heuristic to guess file types works: http://eli.thegreenplace.net/2011/10/19/perls-guess-if-file-is-text-or-binary-implemented-in-python/

And that's just where we started. Over time, we've found more edge cases and our heuristic has gotten more complex.

Also, this package saves you from having to write and thoroughly test your code with all sorts of weird file types and encodings, cross-platform.

Builds

Linux (Ubuntu 12.04 LTS Server Edition 64 bit): Windows (Windows Server 2012 R2 (x64)):

Credits

- Special thanks to Eli Bendersky (@eliben) for his writeup explaining the heuristic and his implementation, which this is largely based on.
- Source code from the portion of Perl's *pp_fttext* that checks for textiness: https://github.com/Perl/perl5/blob/v5.23.1/pp_sys.c#L3527-L3587

CHAPTER 2

Installation

At the command line:

```
$ easy_install binaryornot
```

Or, if you have *pip*:

```
$ pip install binaryornot
```

CHAPTER 3

Quickstart

To use BinaryOrNot in a project, import it and use *is_binary()* to guess whether a file is binary or text.

For example:

```
>>> from binaryornot.check import is_binary
>>> is_binary('README.rst')
False
```

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/audreyr/binaryornot/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/audreyr/binaryornot/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 :)

Get Started!

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

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

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

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/audreyr/binaryornot/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

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

CHAPTER 5

Credits

Development Lead

- Audrey Roy Greenfeld (@audreyr)

Contributors

- Nick Coghlan (@ncoghlan)
- Ville Skyttä (@scop)
- Vincent Bernat (@vincentbernat)
- Daniel Roy Greenfeld (@pydanny)
- Philippe Ombredanne (@pombredanne)
- Aaron Meurer (@asmeurer)
- David R. MacIver (@DRMacIver)
- Raphael Pierzina (@hackebrot)
- Johannes (@johtso)
- Luke Hinds (@lukehinds)

CHAPTER 6

History

0.4.4 (2017-04-13)

- Notify users for file i/o issues. Thanks @lukehinds!

0.4.3 (2017-04-13)

- Restricted chardet to anything 3.0.2 or higher due to https://github.com/chardet/chardet/issues/113. Thanks @dan-blanchard for the quick fix!

0.4.2 (2017-04-12)

- Restricted chardet to anything under 3.0 due to https://github.com/chardet/chardet/issues/113
- Added pyup badge
- Added utilities for pushing new versions up

0.4.0 (2015-08-21)

- Enhanced detection for some binary streams and UTF texts. (#10, 11) Thanks @pombredanne.
- Set up Appveyor for continuous testing on Windows. Thanks @pydanny.
- Update link to Perl source implementation. (#9) Thanks @asmeurer @pombredanne @audreyr.
- Handle UnicodeDecodeError in check. (#12) Thanks @DRMacIver.
- Add very simple Hypothesis based tests. (#13) Thanks @DRMacIver.
- Use setup to determine requirements and remove redundant requirements.txt. (#14) Thanks @hackebrot.

- Add documentation status badge to README.rst. (#15) Thanks @hackebrot.
- Run tox in travis.yml. Add pypy and Python 3.4 to tox environments. (#16) Thanks @hackebrot @pydanny.
- Handle LookupError when detecting encoding. (#17) Thanks @DRMacIver.

0.3.0 (2014-05-05)

- Include tests, docs in source package. (#6) Thanks @vincentbernat.
- Drop unnecessary shebangs and executable bits. (#8) Thanks @scop.
- Generate string of printable extended ASCII bytes only once. (#7) Thanks @scop.
- Make number of bytes to read parametrizable. (#7) Thanks @scop.

0.2.0 (2013-09-22)

- Complete rewrite of everything. Thanks @ncoghlan.

0.1.1 (2013-08-17)

- Tests pass under Python 2.6, 2.7, 3.3, PyPy.

0.1.0 (2013-08-17)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search