spinrewriter Documentation

Release 0.1.6.dev0

April 29, 2014

Spin Rewriter is an online service for spinning text (synonym substitution) that creates unique version(s) of existing text. This package provides a way to easily interact with SpinRewriter API. Usage requires an account, get one here.

- Source code @ GitHub
- Releases @ PyPI
- Documentation @ ReadTheDocs
- Continuous Integration @ Travis-CI

CHAPTER 1

Install

Install into your Python path using *pip* or *easy_install*:

```
$ pip install spinrewriter
$ easy_install spinrewriter
```

CHAPTER 2

Usage

After installing it, this is how you use it:

```
Initialize SpinRewriter.
>>> text = u"This is the text we want to spin."
>>> from spinrewriter import SpinRewriter
>>> rewriter = SpinRewriter('username', 'api_key')

Request processed spun text with spintax.
>>> rewriter.text_with_spintax(text)
u"{This is|This really is|That is|This can be} some text that we'd {like to
|prefer to|want to|love to} spin."

Request a unique variation of processed given text.
>>> rewriter.unique_variation(text)
u"This really is some text that we'd love to spin."
```

CHAPTER 3

API

3.1 API

3.1.1 SpinRewriter

class spinrewriter.**SpinRewriter**(*email_address*, *api_key*)
A facade for easier usage of the raw Spin Rewriter API.

text_with_spintax(*text*, *confidence_level='medium'*)
Return text with spintax elements inserted.

Parameters

- **text** (*string*) – text to process
- **confidence_level** (*Api.CONFIDENCE_LVL*) – how 'confident' the spinner API is when transforming the text

Returns original text with spintax elements

Return type string

unique_variation(*text*, *confidence_level='medium'*)
Return unique variation of the given text.

Parameters

- **text** (*string*) – text to process
- **confidence_level** (*Api.CONFIDENCE_LVL*) – how 'confident' the spinner API is when transforming the text

Returns spinned version of the original text

Return type string

3.1.2 Raw API access

class spinrewriter.**Api**(*email_address*, *api_key*)
A class representing the Spin Rewriter API (http://www.spinrewriter.com/).

ACTION = ACTION(api_quota='api_quota', text_with_spintax='text_with_spintax', unique_variation='unique_variation
collection of possible values for the action parameter

CONFIDENCE_LVL = CONFIDENCE_LVL(low='low', medium='medium', high='high')
collection of possible values for the confidence_level parameter

REQ_P_NAMES = REQ_P_NAMES(email_address='email_address', api_key='api_key', action='action', text='text', prote
collection of all request parameters' names

RESP_P_NAMES = RESP_P_NAMES(status='status', response='response', api_requests_made='api_requests_made', api
collection of all response fields' names

SPINTAX_FORMAT = SPINTAX_FORMAT(pipe_curly='{|}', tilde_curly='{~}', pipe_square='[|]', spin_tag='[spin]')
collection of possible values for the spintax_format parameter

STATUS = STATUS(ok='OK', error='ERROR')
possible response status strings returned by API

URL = 'http://www.spinrewriter.com/action/api'
URL for invoking the API

_raise_error(*api_response*)
Examine the API response and raise exception of the appropriate type.

NOTE: usage of this method only makes sense when API response's status indicates an error

Parameters **api_response** (*dictionary*) – API's response fileds

_send_request(*params*)
Invoke Spin Rewriter API with given parameters and return its response.

Parameters **params** (*tuple of 2-tuples*) – parameters to pass along with the request

Returns API's response (already JSON-decoded)

Return type dictionary

_transform_plain_text(*action*, *text*, *protected_terms*, *confidence_level*, *nested_spintax*, *spintax_format*)
Transform plain text using SpinRewriter API.

Pack parameters into format as expected by the _send_request method and invoke the action method to get transformed text from the API.

Parameters

- **action** (*string*) – name of the action that will be requested from API
- **text** (*string*) – text to process
- **protected_terms** (*list of strings*) – keywords and key phrases that should be left intact
- **confidence_level** (*string*) – the confidence level of the One-Click Rewrite process
- **nested_spintax** (*boolean*) – whether or not to also spin single words inside already spun phrases
- **spintax_format** (*string*) – spintax format to use in returned text

Returns processed text and some other meta info

Return type dictionary

api_quota()
Return the number of made and remaining API calls for the 24-hour period.

Returns remaining API quota

Return type dictionary

text_with_spintax(*text*, *protected_terms=None*, *confidence_level='medium'*, *nested_spintax=False*, *spintax_format='{|}'*)

Return processed spun text with spintax.

Parameters

- **text** (*string*) – original text that needs to be changed
- **protected_terms** (*list of strings*) – (optional) keywords and key phrases that should be left intact
- **confidence_level** (*string*) – (optional) the confidence level of the One-Click Rewrite process
- **nested_spintax** (*boolean*) – (optional) whether or not to also spin single words inside already spun phrases
- **spintax_format** (*string*) – (optional) spintax format to use in returned text

Returns processed text and some other meta info

Return type dictionary

unique_variation(*text*, *protected_terms=None*, *confidence_level='medium'*, *nested_spintax=False*, *spintax_format='{|}'*)

Return a unique variation of the given text.

Parameters

- **text** (*string*) – original text that needs to be changed
- **protected_terms** (*list of strings*) – (optional) keywords and key phrases that should be left intact
- **confidence_level** (*string*) – (optional) the confidence level of the One-Click Rewrite process
- **nested_spintax** (*boolean*) – (optional) whether or not to also spin single words inside already spun phrases
- **spintax_format** (*string*) – (optional) (probably not relevant here? But API documentation not clear here ...)

Returns processed text and some other meta info

Return type dictionary

unique_variation_from_spintax(*text*, *nested_spintax=False*, *spintax_format='{|}'*)

Return a unique variation of an already spun text.

Parameters

- **text** (*string*) – text to process
- **nested_spintax** (*boolean*) – whether or not to also spin single words inside already spun phrases
- **spintax_format** (*string*) – (probably not relevant here? But API documentation not clear here ...)

Returns processed text and some other meta info

Return type dictionary

3.1.3 Exceptions

exception spinrewriter.exceptions.**AuthenticationError**(*api_error_msg*)
Raised when authentication error occurs.

exception spinrewriter.exceptions.**InternalApiError**(*api_error_msg*)
Raised when unexpected error occurs on the API server when processing a request.

exception spinrewriter.exceptions.**MissingParameterError**(*api_error_msg*)
Raised when required parameter is not provided.

exception spinrewriter.exceptions.**ParamValueError**(*api_error_msg*)
Raised when parameter passed to API call has an invalid value.

exception spinrewriter.exceptions.**QuotaLimitError**(*api_error_msg*)
Raised when API quota limit is reached.

exception spinrewriter.exceptions.**SpinRewriterApiError**(*api_error_msg*)
Base class for exceptions in Spin Rewriter module.

exception spinrewriter.exceptions.**UnknownActionError**(*api_error_msg*)
Raised when unknown API action is requested.

exception spinrewriter.exceptions.**UnknownApiError**(*api_error_msg*)
Raised when API call results in an unrecognized error.

exception spinrewriter.exceptions.**UsageFrequencyError**(*api_error_msg*)
Raised when subsequent API requests are made in a too short time interval.

CHAPTER 4

Developer documentation

4.1 Developer environment

4.1.1 Dependencies

A C/C++ compilation environment

On a Debian based system install the `build-essential` package. On OS X, install XCode and MacPorts.

Git

On a Debian based system install the `git-core` package. On OS X, get the latest version from http://code.google.com/p/git-osx-installer/.

Python 2.7

On a Debian based system install the `python2.7-dev` package. On OS X (and others) use the buildout.python to prepare a clean Python installation.

Virtualenv

Recommended installation in virtualenv.

4.1.2 Build

First, you need to *clone* the git repository on GitHub to download the code to your local machine:

```
$ git clone git@github.com:niteoweb/spinrewriter.git
```

What follows is initializing the *buildout* environment:

```
$ cd spinrewriter
$ virtualenv .
$ bin/python bootstrap.py
```

And now you can *run the buildout*. This will fetch and configure tools and libs needed for developing *spinrewriter*:

```
$ bin/buildout
```

4.1.3 Verify

Your environment should now be ready. Test that by using the `py` Python interpreter inside the `bin` directory, which has *spinrewriter* installed in it's path:

```
$ bin/py

>>> from spinrewriter import SpinRewriter
>>> rewriter = SpinRewriter('username', 'api_key')
>>> rewriter.unique_variation('Some random text.')
u"Some random lines."
```

The code for *spinrewriter* lives in `src/`. Make a change and re-run `bin/py` to see it resembled!

Moreover, you should have the following tools in the `bin/` directory, ready for use:

pyflakes

A sintax validation tool.

pep8

A sintax validation tool.

sphinbuilder

A tool for testing HTML render of *spinrewriter*'s documentation.

longtest

A tool for testing the HTML render of the package description (part of `zest.releaser`).

mkrelease

A tool we use for releasing a new version (part of `jarn.mkrelease`).

4.2 Conventions

4.2.1 Line length

All Python code in this package should be PEP8 valid. However, we don't enforce the 80-char line length rule. It is encouraged to have your code formatted in 80-char lines, but somewhere it's just more readable to break this rule for a few characters. Long and descriptive test method names are a good example of this.

Note: Configuring your editor to display a line at 80th column helps a lot here and saves time.

Note: The line length rules also applies to non-python source files, such as documentation .rst files.

4.2.2 About imports

1. Don't use * to import *everything* from a module.
2. Don't use commas to import multiple stuff on a single line.
3. Don't use relative paths.

```
from collective.table.local import add_row
from collective.table.local import delete_rows
from collective.table.local import update_cell
```

instead of

```
from collective.table.local import *
from collective.table.local import add_row, delete_rows
from .local import update_cell
```

4.2.3 Sort imports

As another imports stylistic guide: Imports of code from other modules should always be alphabetically sorted with no empty lines between imports. The only exception to this rule is to keep one empty line between a group of `from x import y` and a group of `import y` imports.

```
from collective.table.tests.base import TableIntegrationTestCase
from plone.app.testing import login

import os
```

instead of

```
import os

from plone.app.testing import login
from collective.table.tests.base import TableIntegrationTestCase
```

4.2.4 Commit checklist

Before every commit you should:

- Run *Unit tests*.
- Run *Syntax validation*.
- Add an entry to *Changelog* (if applicable).
- Add/modify *Sphinx documentation* (if applicable).

Note: All syntax checks and all tests can be run with a single command:

```
$ ./pre-commit-check.sh
```

4.2.5 Unit tests

Un-tested code is broken code.

For every feature you add to the codebase you must also add tests for it. Also write a test for every bug you fix to ensure it doesn't crop up again in the future.

You run tests like this:

```
$ bin/test
```

4.2.6 Syntax validation

All Python source code should be *PEP-8* valid and checked for syntax errors. Tools for checking this are *pep8* and *pyflakes*.

To validate your source code, run the following two commands:

```
$ bin/pyflakes src/spinrewriter
$ bin/pep8 --ignore=E501 src/spinrewriter
```

Note: It pays off to invest a little time to make your editor run *pep8* and *pyflakes* on a file every time you save that file. Saves lots of time in the long run.

4.2.7 Changelog

Feature-level changes to code are tracked inside `docs/HISTORY.txt`. Examples:

- added feature X
- removed Y
- fixed bug Z

Add an entry every time you add/remove a feature, fix a bug, etc.

4.2.8 Sphinx documentation

Un-documented code is broken code.

For every feature you add to the codebase you should also add documentation for it to `docs/`.

After adding/modifying documentation, re-build *Sphinx* and check how it is displayed:

```
$ bin/sphinxbuilder
$ open docs/html/index.html
```

Documentation is automatically generated from these source files every time you push your code to GitHub. The post-commit hook is handled by ReadTheDocs and the results are visible at http://readthedocs.org/docs/spinrewriter.

4.3 Releasing a new version

Releasing a new version of *spinrewriter* involves the following steps:

1. Create a git tag for the release.
2. Push the git tag upstream to GitHub.
3. Generate a distribution file for the package.
4. Upload the generated package to Python Package Index (PyPI).

4.3.1 Checklist

Before every release make sure that:

1. You have documented your changes in the `HISTORY.rst` file and set the release date for the version you are releasing.
2. You have modified the version identifier in the `version.txt` to reflect the new release.
3. You have confirmed that the package description (generated from `README.rst` and others) renders correctly by running `bin/longtest`.
4. You have committed all changes to the git repository and pushed them upstream.
5. You have the working directory checked out at the revision you wish to release.

4.3.2 Actions

For help with releasing we use `jarn.mkreleaser`. It's listed as a dependency in `setup.py` and should already be installed in your local bin:

```
$ bin/mkrelease -d pypi -pq ./
```

Note: In order to push packages to PyPI you need to have the appropriate access rights to the package on PyPI and you need to configure your PyPI credentials in the `~/.pypirc` file, e.g.:

```
[distutils]
index-servers =
  pypi

[pypi]
username = fred
password = secret
```

4.3.3 Example

In the following example we are releasing version 0.1 of *spinrewriter*. The package has been prepared so that `version.txt` contains the version `0.1`, this change has been committed to git and all changes have been pushed upstream to GitHub:

```
# Check that package description is rendered correctly
$ bin/longtest

# Make a release and upload it to PyPI
$ bin/mkrelease -d pypi -pq ./
Releasing spinrewriter 0.1
Tagging spinrewriter 0.1
To git@github.com:niteoweb/spinrewriter.git
* [new tag]         0.1 -> 0.1
running egg_info
running sdist
warning: sdist: standard file not found: should have one of README, README.txt
running register
Server response (200): OK
running upload
warning: sdist: standard file not found: should have one of README, README.txt
Server response (200): OK
done
```

Note: Please ignore the sdist warning about README file above. PyPI does not depend on it and it's just a bug in

setupools (reported and waiting to be fixed).

CHAPTER 5

Credits

- Development by NiteoWeb Ltd.
- Code and documentation snippets inspired by HexagonIT Oy.
- Similar package used as point-of-reference is thebestspinner by Peter Flood.

5.1 Changelog

5.1.1 0.1.6 (unreleased)

- Replace development tools and 100% test coverage. [matejc]

5.1.2 0.1.5 (2012-12-17)

- Added tests for parsing error messages and fixed loads of nasty bugs found while writing these tests. [zupo]

5.1.3 0.1.4 (2012-11-06)

- Fixed spelling error in one of authentication error's message. [zupo]
- Use 'vvv' for syntax validation. [matejc]

5.1.4 0.1.3 (2012-07-31)

- Fixed unicode encode/decode error for article texts containing non-ascii characters. [plamut]
- Added missing test coverage for the unique_variation_from_spintax method. [plamut]

5.1.5 0.1.2 (2012-07-24)

- Various fixes of bugs that surfaced when lib was put into staging. [zupo]

5.1.6 0.1.1 (2012-04-13)

- A URL in README.txt was missing a leading `http` which broke reST rendering on PyPI. [zupo]

5.1.7 0.1 (2012-04-13)

- SpinRewriter facade class. [plamut]
- Tests and documentation. [zupo]
- Raw API access class. [plamut]
- Project skeleton. [zupo]

5.2 License (3-clause BSD)

CHAPTER 6

Indices and tables

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

S