Kotonoha

Release 0.4.1

Oct 17, 2019

Contents

CHAPTER 1

Overview

docs	
tests	
package	

Tools for preprocessing Japanese texts

- Free software: MIT license

CHAPTER 2

Installation

At the command line:

```
pip install kotonoha
```

CHAPTER 3

Usage

To use Kotonoha in a project:

..code-block:

```
from kotonoha import Kotonoha

jtp = Kotonoha()

pipeline = [
        {
                'replace_numbers': {'replace_text': '##'}
        },
        {
                'remove_url'
        }
]

jtp.prepare(pipeline)
jtp.run('9000! http://some.url')  # => '##! '
```

3.1 Getting started

Kotonoha will execute all tasks defined in the pipeline sequentially.

3.2 List of tasks

- *alpha_to_full*: Converts all alphabet words to full-width characters (kore =>).
- *digits*: Converts all numbers to half-width characters (=> 1234).
- *to_full_width*: Converts to full-width characters (=>).

- *lower*: Converts to lower case.
- *replace_numbers*: Replace all numbers with a `replace_text` (default #).
- *remove_numbers*: Remove all numbers.
- *replace_prices*: Replace all prices with a `replace_text` (default #). Prices should have the format 1,234,567.8912.
- *remove_prices*: Remove all prices.
- *replace_url*: Replace all urls with a `replace_text` (default ‘’).
- *remove_url*: Remove all urls.
- *replace_hashtags*: Replace all hashtags with a `replace_text` (default ‘’).
- *replace_emails*: Replace all emails with a `replace_text` (default ‘’).
- *replace_mentions*: Replace all mentions with a `replace_text` (default ‘’).

3.3 MeCab handler

There is a class MeCabHandler which can be used to simplify some basic configurations for filtering and lemmatization of words.

```
from kotonoha import MeCabHandler
import MeCab

tagger = MeCab.Tagger('-Ochasen -d ' + neologd_path)

handler = MeCabHandler(tagger)

handler.nouns('...')  # => string containing nouns, separated by spaces, all words␣
↪are in their lemma format.
handler.verbs('...')  # => string containing verbs, separated by spaces, all words␣
↪are in their lemma format.
handler.meaningful('...')  # => string containing nouns, verbs and adjectives,␣
↪separated by spaces, all words are in their lemma format.
handler.basic('...')  # => string containing all words, separated by spaces, all␣
↪words are in their lemma format.
```

If you need to use a custom filter for MeCab, you can use the `by_filter` function and implement your own custom filter function. The filter function will receive a list of 8 strings containing the 7 features from MeCab’s parseToNode result + the surface. The filter function must return a text.

..code-block:: python

    from kotonoha import MeCabHandler import MeCab

    tagger = MeCab.Tagger(‘-Ochasen -d ‘ + neologd_path)

    handler = MeCabHandler(tagger)

    def my_custom_filter(args):

        if args[0] == ‘’: return args[6]

        if args[0] == ‘’ and args[1] == ‘’: return args[6]

        if args[0] == ‘’ and args[1] not in {‘’, ‘’}: return args[8]

        else return ‘’

handler.by_filter('. . . ', my_custom_filter)

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

4.1 Bug reports

When 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.

4.2 Documentation improvements

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

4.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/brunotoshio/kotonoha/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 code contributions are welcome :)

4.4 Development

To set up *kotonoha* for local development:

1. Fork kotonoha (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/kotonoha.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

4.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

4.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 5

Authors

- Bruno Toshio Sugano

CHAPTER 6

Changelog

6.1 0.4.1 (2019-09-05)

- Small fix in by_filter

6.2 0.4.0 (2019-09-05)

- Added a new method in MeCabHandler: by_filter

6.3 0.3.1 (2019-05-24)

- Small fixes

6.4 0.3.0 (2019-05-24)

- Replace Hashtags
- Replace Mentions
- Replace Emails

6.5 0.2.1 (2019-04-26)

- Small fix in MeCabHandler

6.6 0.1.0 (2019-04-24)

- Added MeCabHandler

6.7 0.0.0 (2019-04-23)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search