

---

# Monogatari

*Release 1.3.0*

Jun 18, 2019



---

## Contents

---

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>Usage</b>	<b>5</b>
3.1	Available dictionaries . . . . .	6
<b>4</b>	<b>Contributing</b>	<b>7</b>
4.1	Bug reports . . . . .	7
4.2	Documentation improvements . . . . .	7
4.3	Feature requests and feedback . . . . .	7
4.4	Development . . . . .	8
<b>5</b>	<b>Authors</b>	<b>9</b>
<b>6</b>	<b>Changelog</b>	<b>11</b>
6.1	1.3.0 (2019-06-18) . . . . .	11
6.2	1.2.0 (2019-04-26) . . . . .	11
6.3	1.1.0 (2019-04-26) . . . . .	11
6.4	1.0.0 (2019-04-26) . . . . .	11
6.5	0.0.1 (2019-04-26) . . . . .	11
6.6	0.0.0 (2019-04-26) . . . . .	11



# CHAPTER 1

---

## Overview

---

docs	
tests	
package	

Tool for handling dictionaries.

- Free software: MIT license



## CHAPTER 2

---

### Installation

---

At the command line:

```
pip install monogatari
```





## CHAPTER 3

---

### Usage

---

To use monogatari in a project:

```
from monogatari import JMFDCounter

counter = JMFDCounter()

list_of_words = ['', '', '', '']

counter.count(list_of_words)

counter.top(100) # List top N categories, ordered by number of words

counter.top_normalized(100) # List top N categories, ordered by number of words,
↪normalized by the total number of words
```

In this example, we are using J-MFD dictionary (<https://github.com/soramame0518/j-mfd>) as the base for word counting. However, we can load a custom dictionary using the class `DictCounter`:

```
from monogatari import DictCounter

counter = DictCounter('/path/to/the/dictionary.dic')

list_of_words = ['', '', '', '']

counter.count(list_of_words)

counter.top(100) # List top N categories, ordered by number of words

counter.top_normalized(100) # List top N categories, ordered by number of words,
↪normalized by the total number of words
```

The dictionary file must have the following structure:

```
%
Category_key_1    Category_value_1
Category_key_2    Category_value_2
Category_key_3    Category_value_3
Category_key_4    Category_value_4
%

Word_key_1        Category_key_1  Category_key_2
Word_key_2        Category_key_1
Word_key_3        Category_key_2
Word_key_4        Category_key_1  Category_key_3
Word_key_5        Category_key_1  Category_key_2  Category_key_4
Word_key_6        Category_key_4
```

## 3.1 Available dictionaries

- MFD (from <https://www.moralfoundations.org/>):

```
from monogatari import MFDCounter

counter = MFDCounter()
```

- JMFD (from <https://github.com/soramame0518/j-mfd>):

```
from monogatari import JMFDCounter

counter = JMFDCounter()
```

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

monogatari could always use more documentation, whether as part of the official monogatari 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/monogatari/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 *monogatari* for local development:

1. Fork [monogatari](#) (look for the “Fork” button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/monogatari.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`)<sup>1</sup>.
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
```

---

<sup>1</sup> 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



### **6.1 1.3.0 (2019-06-18)**

- Small fixes

### **6.2 1.2.0 (2019-04-26)**

- Words found in each category

### **6.3 1.1.0 (2019-04-26)**

- MFD dictionary added

### **6.4 1.0.0 (2019-04-26)**

- JMFD added.

### **6.5 0.0.1 (2019-04-26)**

- DictHandler

### **6.6 0.0.0 (2019-04-26)**

- First release on PyPI.