1 安装

```
pip install flashtext
```

2 提取关键字

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> # keyword_processor.add_keyword(<unclean name>, <standardised name>)
>>> keyword_processor.add_keyword('Big Apple', 'New York')
>>> keyword_processor.add_keyword('Bay Area')
>>> keywords_found = keyword_processor.extract_keywords('I love Big Apple and Bay
↪ Area.')
>>> keywords_found
>>> # ['New York']
```

3 替换关键字

```
>>> keyword_processor.add_keyword('New Delhi', 'NCR region')
>>> new_sentence = keyword_processor.replace_keywords('I love Big Apple and new
↪ delhi.')
>>> new_sentence
>>> # 'I love New York and NCR region.'
```

3.1 Case Sensitive example

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor(case_sensitive=True)
>>> keyword_processor.add_keyword('Big Apple', 'New York')
>>> keyword_processor.add_keyword('Bay Area')
>>> keywords_found = keyword_processor.extract_keywords('I love big Apple and Bay
↪ Area.')
>>> keywords_found
>>> # ['Bay Area']
```

3.2 关键字空格信息

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_processor.add_keyword('Big Apple', 'New York')
>>> keyword_processor.add_keyword('Bay Area')
>>> keywords_found = keyword_processor.extract_keywords('I love big Apple and Bay
↪  Area.', span_info=True)
>>> keywords_found
>>> # [('New York', 7, 16), ('Bay Area', 21, 29)]
```

3.3 用关键字提取额外的信息

```
>>> from flashtext import KeywordProcessor
>>> kp = KeywordProcessor()
>>> kp.add_keyword('Taj Mahal', ('Monument', 'Taj Mahal'))
>>> kp.add_keyword('Delhi', ('Location', 'Delhi'))
>>> kp.extract_keywords('Taj Mahal is in Delhi.')
>>> # [('Monument', 'Taj Mahal'), ('Location', 'Delhi')]
>>> # NOTE: replace_keywords feature won't work with this.
```

No clean name for Keywords

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_processor.add_keyword('Big Apple')
>>> keyword_processor.add_keyword('Bay Area')
>>> keywords_found = keyword_processor.extract_keywords('I love big Apple and Bay
↪  Area.')
>>> keywords_found
>>> # ['Big Apple', 'Bay Area']
```

3.4 同时添加多个关键字

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_dict = {
```

```
>>>     "java": ["java_2e", "java programing"],
>>>     "product management": ["PM", "product manager"]
>>> }
>>> # {'clean_name': ['list of unclean names']}
>>> keyword_processor.add_keywords_from_dict(keyword_dict)
>>> # Or add keywords from a list:
>>> keyword_processor.add_keywords_from_list(["java", "python"])
>>> keyword_processor.extract_keywords('I am a product manager for a java_2e
↪ platform')
>>> # output ['product management', 'java']
```

3.5 移除关键字

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_dict = {
>>>     "java": ["java_2e", "java programing"],
>>>     "product management": ["PM", "product manager"]
>>> }
>>> keyword_processor.add_keywords_from_dict(keyword_dict)
>>> print(keyword_processor.extract_keywords('I am a product manager for a java_2e
↪ platform'))
>>> # output ['product management', 'java']
>>> keyword_processor.remove_keyword('java_2e')
>>> # you can also remove keywords from a list/ dictionary
>>> keyword_processor.remove_keywords_from_dict({"product management": ["PM"]})
>>> keyword_processor.remove_keywords_from_list(["java programing"])
>>> keyword_processor.extract_keywords('I am a product manager for a java_2e
↪ platform')
>>> # output ['product management']
```

3.6 To check Number of terms in KeywordProcessor

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_dict = {
>>>     "java": ["java_2e", "java programing"],
```

```
>>>     "product management": ["PM", "product manager"]
>>> }
>>> keyword_processor.add_keywords_from_dict(keyword_dict)
>>> print(len(keyword_processor))
>>> # output 4
```

3.7 To check if term is present in KeywordProcessor

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_processor.add_keyword('j2ee', 'Java')
>>> 'j2ee' in keyword_processor
>>> # output: True
>>> keyword_processor.get_keyword('j2ee')
>>> # output: Java
>>> keyword_processor['colour'] = 'color'
>>> keyword_processor['colour']
>>> # output: color
```

3.8 Get all keywords in dictionary

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_processor.add_keyword('j2ee', 'Java')
>>> keyword_processor.add_keyword('colour', 'color')
>>> keyword_processor.get_all_keywords()
>>> # output: {'colour': 'color', 'j2ee': 'Java'}
```

For detecting Word Boundary currently any character other than this is considered a word boundary.

To set or add characters as part of word characters

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> keyword_processor.add_keyword('Big Apple')
>>> print(keyword_processor.extract_keywords('I love Big Apple/Bay Area.'))
>>> # ['Big Apple']
```

```
>>> keyword_processor.add_non_word_boundary('/')
>>> print(keyword_processor.extract_keywords('I love Big Apple/Bay Area.'))
>>> # []
```

API doc KeywordProcessor

API Doc Import and initialize module Add Keywords to module Extract keywords Replace keywords Add keywords from File Add keywords from dict Add keywords from list KeywordProcessor Class Doc Test

```
$ git clone https://github.com/vi3k6i5/flashtext
$ cd flashtext
$ pip install pytest
$ python setup.py test
```

Build Docs

```
$ git clone https://github.com/vi3k6i5/flashtext
$ cd flashtext/docs
$ pip install sphinx
$ make html
$ # open _build/html/index.html in browser to view it locally
```

Why not Regex? It' s a custom algorithm based on Aho-Corasick algorithm and Trie Dictionary.

Benchmark Time taken by FlashText to find terms in comparison to Regex.

https://thepracticaldev.s3.amazonaws.com/i/xruf50n6z1r37ti8rd89.png Time taken by FlashText to replace terms in comparison to Regex.

https://thepracticaldev.s3.amazonaws.com/i/k44ghwp8o712dm58debj.png Link to code for benchmarking the Find Feature and Replace Feature.

The idea for this library came from the following StackOverflow question.

Citation The original paper published on FlashText algorithm.

4 flashtext API

4.1 Import and initialize module

```
>>> from flashtext import KeywordProcessor
>>> keyword_processor = KeywordProcessor()
>>> # if match has to be case sensitive
>>> keyword_processor = KeywordProcessor(case_sensitive=True)
```

4.2 Add Keywords to module

```
>>> keyword_processor.add_keyword('Big Apple', 'New York')
>>> keyword_processor.add_keyword('Bay Area')
Extract keywords
>>> keywords_found = keyword_processor.extract_keywords('I love Big Apple and Bay Area.')
>>> keywords_found
>>> ['New York', 'Bay Area']
```

4.3 Replace keywords

```
>>> keyword_processor.add_keyword('New Delhi', 'NCR region')
>>> new_sentence = keyword_processor.replace_keywords('I love Big Apple and new delhi.')
>>> new_sentence
>>> 'I love New York and NCR region.'
```

4.4 Add keywords from File

```
>>> # Option 1: keywords.txt content
>>> # java_2e=>java
>>> # java programing=>java
>>> # product management=>product management
>>> # product management techniques=>product management
>>> # Option 2: keywords.txt content
>>> # java
>>> # python
>>> # c++
>>> keyword_processor.add_keyword_from_file('keywords.txt')
```

4.5 Add keywords from dict

```
>>> keyword_dict = {
        "java": ["java_2e", "java programing"],
        "product management": ["PM", "product manager"]
```

```
    }
>>> keyword_processor.add_keywords_from_dict(keyword_dict)
```

4.6 Add keywords from list

```
>>> keyword_processor.add_keywords_from_list(["java", "python"]})
```