This repository Search Pull requests Issues Gist

fxsjy / **jieba**

Watch 823 Star 8,320 Fork 2,537

Code Issues 260 Pull requests 17 Projects 0 Wiki Pulse Graphs

结巴中文分词

488 commits **2** branches **23** releases **32** contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

fxsjy committed on **GitHub** Merge pull request #455 from OOCZC/master … Latest commit `4eef868` 26 days ago

extra_dict	update to v0.33	3 years ago
jieba	Bugfix for HMM=False in parallelism.	9 months ago
test	Bugfix for HMM=False in parallelism.	9 months ago
.gitattributes	first commit	5 years ago
.gitignore	update jieba3k	3 years ago
Changelog	version change 0.38	a year ago
LICENSE	add a license file	4 years ago
MANIFEST.in	include Changelog & README.md in the distribution package	4 years ago
README.md	Update README.md	28 days ago
setup.py	version change 0.38	a year ago

README.md

jieba

“结巴”中文分词：做最好的 Python 中文分词组件

"Jieba" (Chinese for "to stutter") Chinese text segmentation: built to be the best Python Chinese word segmentation module.

- *Scroll down for English documentation.*

特点

- 支持三种分词模式：
  - 精确模式，试图将句子最精确地切开，适合文本分析；
  - 全模式，把句子中所有的可以成词的词语都扫描出来, 速度非常快，但是不能解决歧义；
  - 搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。
- 支持繁体分词
- 支持自定义词典
- MIT 授权协议

在线演示

http://jiebademo.ap01.aws.af.cm/

(Powered by Appfog)

网站代码：https://github.com/fxsjy/jiebademo

安装说明

代码对 Python 2/3 均兼容

- 全自动安装： `easy_install jieba` 或者 `pip install jieba` / `pip3 install jieba`
- 半自动安装：先下载 http://pypi.python.org/pypi/jieba/ ，解压后运行 `python setup.py install`
- 手动安装：将 jieba 目录放置于当前目录或者 site-packages 目录
- 通过 `import jieba` 来引用

算法

- 基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图 (DAG)
- 采用了动态规划查找最大概率路径, 找出基于词频的最大切分组合
- 对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法

主要功能

1. 分词

- `jieba.cut` 方法接受三个输入参数: 需要分词的字符串；cut_all 参数用来控制是否采用全模式；HMM 参数用来控制是否使用 HMM 模型
- `jieba.cut_for_search` 方法接受两个参数：需要分词的字符串；是否使用 HMM 模型。该方法适合用于搜索引擎构建倒排索引的分词，粒度比较细
- 待分词的字符串可以是 unicode 或 UTF-8 字符串、GBK 字符串。注意：不建议直接输入 GBK 字符串，可能无法预料地错误解码成 UTF-8
- `jieba.cut` 以及 `jieba.cut_for_search` 返回的结构都是一个可迭代的 generator，可以使用 for 循环来获得分词后得到的每一个词语(unicode)，或者用
- `jieba.lcut` 以及 `jieba.lcut_for_search` 直接返回 list
- `jieba.Tokenizer(dictionary=DEFAULT_DICT)` 新建自定义分词器，可用于同时使用不同词典。 `jieba.dt` 为默认分词器，所有全局分词相关函数都是该分词器的映射。

代码示例

```
# encoding=utf-8
import jieba

seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
print("Full Mode: " + "/ ".join(seg_list))  # 全模式

seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("Default Mode: " + "/ ".join(seg_list))  # 精确模式

seg_list = jieba.cut("他来到了网易杭研大厦")  # 默认是精确模式
print(", ".join(seg_list))

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 搜索引擎模式
print(", ".join(seg_list))
```

输出:

```
【全模式】: 我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学

【精确模式】: 我/ 来到/ 北京/ 清华大学

【新词识别】：他, 来到, 了, 网易, 杭研, 大厦    (此处，“杭研”并没有在词典中，但是也被Viterbi算法识别出来了)

【搜索引擎模式】： 小明, 硕士, 毕业, 于, 中国, 科学, 学院, 科学院, 中国科学院, 计算, 计算所, 后, 在, 日本, 京都, 大学, 日本京
```

2. 添加自定义词典

载入词典

- 开发者可以指定自己自定义的词典，以便包含 jieba 词库里没有的词。虽然 jieba 有新词识别能力，但是自行添加新词可以保证更高的正确率
- 用法： jieba.load_userdict(file_name) # file_name 为文件类对象或自定义词典的路径
- 词典格式和 `dict.txt` 一样，一个词占一行；每一行分三部分：词语、词频（可省略）、词性（可省略），用空格隔开，顺序不可颠倒。 `file_name` 若为路径或二进制方式打开的文件，则文件必须为 UTF-8 编码。
- 词频省略时使用自动计算的能保证分出该词的词频。

例如：

```
创新办 3 i
云计算 5
凯特琳 nz
台中
```

- 更改分词器（默认为 `jieba.dt` ）的 `tmp_dir` 和 `cache_file` 属性，可分别指定缓存文件所在的文件夹及其文件名，用于受限的文件系统。
- 范例：
    - 自定义词典：https://github.com/fxsjy/jieba/blob/master/test/userdict.txt
    - 用法示例：https://github.com/fxsjy/jieba/blob/master/test/test_userdict.py
        - 之前： 李小福 / 是 / 创新 / 办 / 主任 / 也 / 是 / 云 / 计算 / 方面 / 的 / 专家 /
        - 加载自定义词库后： 李小福 / 是 / 创新办 / 主任 / 也 / 是 / 云计算 / 方面 / 的 / 专家 /

调整词典

- 使用 `add_word(word, freq=None, tag=None)` 和 `del_word(word)` 可在程序中动态修改词典。
- 使用 `suggest_freq(segment, tune=True)` 可调节单个词语的词频，使其能（或不能）被分出来。
- 注意：自动计算的词频在使用 HMM 新词发现功能时可能无效。

代码示例：

```
>>> print('/'.join(jieba.cut('如果放到post中将出错。', HMM=False)))
如果/放到/post/中将/出错/。
>>> jieba.suggest_freq(('中', '将'), True)
494
>>> print('/'.join(jieba.cut('如果放到post中将出错。', HMM=False)))
如果/放到/post/中/将/出错/。
>>> print('/'.join(jieba.cut('「台中」正确应该不会被切开', HMM=False)))
「/台/中/」/正确/应该/不会/被/切开
>>> jieba.suggest_freq('台中', True)
69
>>> print('/'.join(jieba.cut('「台中」正确应该不会被切开', HMM=False)))
「/台中/」/正确/应该/不会/被/切开
```

- "通过用户自定义词典来增强歧义纠错能力" --- https://github.com/fxsjy/jieba/issues/14

3. 关键词提取

基于 TF-IDF 算法的关键词抽取

`import jieba.analyse`

- jieba.analyse.extract_tags(sentence, topK=20, withWeight=False, allowPOS=())
    - sentence 为待提取的文本
    - topK 为返回几个 TF/IDF 权重最大的关键词，默认值为 20

  - withWeight 为是否一并返回关键词权重值，默认值为 False
  - allowPOS 仅包括指定词性的词，默认值为空，即不筛选
- jieba.analyse.TFIDF(idf_path=None) 新建 TFIDF 实例，idf_path 为 IDF 频率文件

代码示例 （关键词提取）

https://github.com/fxsjy/jieba/blob/master/test/extract_tags.py

关键词提取所使用逆向文件频率（IDF）文本语料库可以切换成自定义语料库的路径

- 用法： jieba.analyse.set_idf_path(file_name) # file_name为自定义语料库的路径
- 自定义语料库示例：https://github.com/fxsjy/jieba/blob/master/extra_dict/idf.txt.big
- 用法示例：https://github.com/fxsjy/jieba/blob/master/test/extract_tags_idfpath.py

关键词提取所使用停止词（Stop Words）文本语料库可以切换成自定义语料库的路径

- 用法： jieba.analyse.set_stop_words(file_name) # file_name为自定义语料库的路径
- 自定义语料库示例：https://github.com/fxsjy/jieba/blob/master/extra_dict/stop_words.txt
- 用法示例：https://github.com/fxsjy/jieba/blob/master/test/extract_tags_stop_words.py

关键词一并返回关键词权重值示例

- 用法示例：https://github.com/fxsjy/jieba/blob/master/test/extract_tags_with_weight.py

基于 TextRank 算法的关键词抽取

- jieba.analyse.textrank(sentence, topK=20, withWeight=False, allowPOS=('ns', 'n', 'vn', 'v')) 直接使用，接口相同，注意默认过滤词性。
- jieba.analyse.TextRank() 新建自定义 TextRank 实例

算法论文： TextRank: Bringing Order into Texts

基本思想:

1. 将待抽取关键词的文本进行分词
2. 以固定窗口大小(默认为5，通过span属性调整)，词之间的共现关系，构建图
3. 计算图中节点的PageRank，注意是无向带权图

使用示例:

见 test/demo.py

4. 词性标注

- `jieba.posseg.POSTokenizer(tokenizer=None)` 新建自定义分词器， `tokenizer` 参数可指定内部使用的 `jieba.Tokenizer` 分词器。 `jieba.posseg.dt` 为默认词性标注分词器。
- 标注句子分词后每个词的词性，采用和 ictclas 兼容的标记法。
- 用法示例

```
>>> import jieba.posseg as pseg
>>> words = pseg.cut("我爱北京天安门")
>>> for word, flag in words:
...    print('%s %s' % (word, flag))
...
我 r
爱 v
北京 ns
天安门 ns
```

5. 并行分词

- 原理：将目标文本按行分隔后，把各行文本分配到多个 Python 进程并行分词，然后归并结果，从而获得分词速度的可观提升

- 基于 python 自带的 multiprocessing 模块，目前暂不支持 Windows
- 用法：
  - `jieba.enable_parallel(4)` # 开启并行分词模式，参数为并行进程数
  - `jieba.disable_parallel()` # 关闭并行分词模式
- 例子：https://github.com/fxsjy/jieba/blob/master/test/parallel/test_file.py
- 实验结果：在 4 核 3.4GHz Linux 机器上，对金庸全集进行精确分词，获得了 1MB/s 的速度，是单进程版的 3.3 倍。
- **注意**：并行分词仅支持默认分词器 `jieba.dt` 和 `jieba.posseg.dt` 。

6. Tokenize：返回词语在原文的起止位置

- 注意，输入参数只接受 unicode
- 默认模式

```
result = jieba.tokenize(u'永和服装饰品有限公司')
for tk in result:
    print("word %s\t\t start: %d \t\t end:%d" % (tk[0],tk[1],tk[2]))
```

```
word 永和                start: 0                end:2
word 服装                start: 2                end:4
word 饰品                start: 4                end:6
word 有限公司            start: 6                end:10
```

- 搜索模式

```
result = jieba.tokenize(u'永和服装饰品有限公司', mode='search')
for tk in result:
    print("word %s\t\t start: %d \t\t end:%d" % (tk[0],tk[1],tk[2]))
```

```
word 永和                start: 0                end:2
word 服装                start: 2                end:4
word 饰品                start: 4                end:6
word 有限                start: 6                end:8
word 公司                start: 8                end:10
word 有限公司            start: 6                end:10
```

7. ChineseAnalyzer for Whoosh 搜索引擎

- 引用： `from jieba.analyse import ChineseAnalyzer`
- 用法示例：https://github.com/fxsjy/jieba/blob/master/test/test_whoosh.py

8. 命令行分词

使用示例： `python -m jieba news.txt > cut_result.txt`

命令行选项（翻译）：

```
使用: python -m jieba [options] filename

结巴命令行界面。

固定参数:
  filename              输入文件

可选参数:
  -h, --help            显示此帮助信息并退出
  -d [DELIM], --delimiter [DELIM]
```

```
                        使用 DELIM 分隔词语，而不是用默认的' / '。
                        若不指定 DELIM，则使用一个空格分隔。
  -p [DELIM], --pos [DELIM]
                        启用词性标注；如果指定 DELIM，词语和词性之间
                        用它分隔，否则用 _ 分隔
  -D DICT, --dict DICT  使用 DICT 代替默认词典
  -u USER_DICT, --user-dict USER_DICT
                        使用 USER_DICT 作为附加词典，与默认词典或自定义词典配合使用
  -a, --cut-all         全模式分词（不支持词性标注）
  -n, --no-hmm          不使用隐含马尔可夫模型
  -q, --quiet           不输出载入信息到 STDERR
  -V, --version         显示版本信息并退出

如果没有指定文件名，则使用标准输入。
```

`--help` 选项输出：

```
$> python -m jieba --help
Jieba command line interface.

positional arguments:
  filename              input file

optional arguments:
  -h, --help            show this help message and exit
  -d [DELIM], --delimiter [DELIM]
                        use DELIM instead of ' / ' for word delimiter; or a
                        space if it is used without DELIM
  -p [DELIM], --pos [DELIM]
                        enable POS tagging; if DELIM is specified, use DELIM
                        instead of '_' for POS delimiter
  -D DICT, --dict DICT  use DICT as dictionary
  -u USER_DICT, --user-dict USER_DICT
                        use USER_DICT together with the default dictionary or
                        DICT (if specified)
  -a, --cut-all         full pattern cutting (ignored with POS tagging)
  -n, --no-hmm          don't use the Hidden Markov Model
  -q, --quiet           don't print loading messages to stderr
  -V, --version         show program's version number and exit

If no filename specified, use STDIN instead.
```

延迟加载机制

jieba 采用延迟加载，`import jieba` 和 `jieba.Tokenizer()` 不会立即触发词典的加载，一旦有必要才开始加载词典构建前缀字典。如果你想手工初始 jieba，也可以手动初始化。

```
import jieba
jieba.initialize()  # 手动初始化（可选）
```

在 0.28 之前的版本是不能指定主词典的路径的，有了延迟加载机制后，你可以改变主词典的路径:

```
jieba.set_dictionary('data/dict.txt.big')
```

例子： https://github.com/fxsjy/jieba/blob/master/test/test_change_dictpath.py

其他词典

1. 占用内存较小的词典文件 https://github.com/fxsjy/jieba/raw/master/extra_dict/dict.txt.small
2. 支持繁体分词更好的词典文件 https://github.com/fxsjy/jieba/raw/master/extra_dict/dict.txt.big

下载你所需要的词典，然后覆盖 jieba/dict.txt 即可；或者用 `jieba.set_dictionary('data/dict.txt.big')`

其他语言实现

结巴分词 Java 版本

作者：piaolingxue 地址：https://github.com/huaban/jieba-analysis

结巴分词 C++ 版本

作者：yanyiwu 地址：https://github.com/yanyiwu/cppjieba

结巴分词 Node.js 版本

作者：yanyiwu 地址：https://github.com/yanyiwu/nodejieba

结巴分词 Erlang 版本

作者：falood 地址：https://github.com/falood/exjieba

结巴分词 R 版本

作者：qinwf 地址：https://github.com/qinwf/jiebaR

结巴分词 iOS 版本

作者：yanyiwu 地址：https://github.com/yanyiwu/iosjieba

结巴分词 PHP 版本

作者：fukuball 地址：https://github.com/fukuball/jieba-php

结巴分词 .NET(C#) 版本

作者：anderscui 地址：https://github.com/anderscui/jieba.NET/

结巴分词 Go 版本

- 作者: wangbin 地址: https://github.com/wangbin/jiebago
- 作者: yanyiwu 地址: https://github.com/yanyiwu/gojieba

系统集成

1. Solr: https://github.com/sing1ee/jieba-solr

分词速度

- 1.5 MB / Second in Full Mode
- 400 KB / Second in Default Mode
- 测试环境: Intel(R) Core(TM) i7-2600 CPU @ 3.4GHz；《围城》.txt

常见问题

1. 模型的数据是如何生成的？

详见： https://github.com/fxsjy/jieba/issues/7

2. “台中”总是被切成“台 中”？（以及类似情况）

P(台中) ＜ P(台)×P(中)，“台中”词频不够导致其成词概率较低

解决方法：强制调高词频

`jieba.add_word('台中')` 或者 `jieba.suggest_freq('台中', True)`

3. “今天天气 不错”应该被切成“今天 天气 不错”？（以及类似情况）

解决方法：强制调低词频

`jieba.suggest_freq(('今天', '天气'), True)`

或者直接删除该词 `jieba.del_word('今天天气')`

4. 切出了词典中没有的词语，效果不理想？

解决方法：关闭新词发现

`jieba.cut('丰田太省了', HMM=False)` `jieba.cut('我们中出了一个叛徒', HMM=False)`

更多问题请点击：https://github.com/fxsjy/jieba/issues?sort=updated&state=closed

修订历史

https://github.com/fxsjy/jieba/blob/master/Changelog

jieba

"Jieba" (Chinese for "to stutter") Chinese text segmentation: built to be the best Python Chinese word segmentation module.

Features

- Support three types of segmentation mode:

1. Accurate Mode attempts to cut the sentence into the most accurate segmentations, which is suitable for text analysis.
2. Full Mode gets all the possible words from the sentence. Fast but not accurate.
3. Search Engine Mode, based on the Accurate Mode, attempts to cut long words into several short words, which can raise the recall rate. Suitable for search engines.

- Supports Traditional Chinese
- Supports customized dictionaries
- MIT License

Online demo

http://jiebademo.ap01.aws.af.cm/

(Powered by Appfog)

Usage

- Fully automatic installation: `easy_install jieba` or `pip install jieba`
- Semi-automatic installation: Download http://pypi.python.org/pypi/jieba/ , run `python setup.py install` after extracting.
- Manual installation: place the `jieba` directory in the current directory or python `site-packages` directory.
- `import jieba` .

Algorithm

- Based on a prefix dictionary structure to achieve efficient word graph scanning. Build a directed acyclic graph (DAG) for all possible word combinations.
- Use dynamic programming to find the most probable combination based on the word frequency.
- For unknown words, a HMM-based model is used with the Viterbi algorithm.

Main Functions

1. Cut

- The `jieba.cut` function accepts three input parameters: the first parameter is the string to be cut; the second parameter is `cut_all`, controlling the cut mode; the third parameter is to control whether to use the Hidden Markov Model.
- `jieba.cut_for_search` accepts two parameter: the string to be cut; whether to use the Hidden Markov Model. This will cut the sentence into short words suitable for search engines.
- The input string can be an unicode/str object, or a str/bytes object which is encoded in UTF-8 or GBK. Note that using GBK encoding is not recommended because it may be unexpctly decoded as UTF-8.
- `jieba.cut` and `jieba.cut_for_search` returns an generator, from which you can use a `for` loop to get the segmentation result (in unicode).
- `jieba.lcut` and `jieba.lcut_for_search` returns a list.
- `jieba.Tokenizer(dictionary=DEFAULT_DICT)` creates a new customized Tokenizer, which enables you to use different dictionaries at the same time. `jieba.dt` is the default Tokenizer, to which almost all global functions are mapped.

Code example: segmentation

```
#encoding=utf-8
import jieba

seg_list = jieba.cut("我来到北京清华大学", cut_all=True)
print("Full Mode: " + "/ ".join(seg_list))  # 全模式

seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("Default Mode: " + "/ ".join(seg_list))  # 默认模式

seg_list = jieba.cut("他来到了网易杭研大厦")
print(", ".join(seg_list))

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 搜索引擎模式
print(", ".join(seg_list))
```

Output:

```
[Full Mode]: 我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学

[Accurate Mode]: 我/ 来到/ 北京/ 清华大学

[Unknown Words Recognize] 他, 来到, 了, 网易, 杭研, 大厦    (In this case, "杭研" is not in the dictionary, but is iden

[Search Engine Mode]:  小明, 硕士, 毕业, 于, 中国, 科学, 学院, 科学院, 中国科学院, 计算, 计算所, 后, 在, 日本, 京都, 大学,
```

2. Add a custom dictionary

Load dictionary

- Developers can specify their own custom dictionary to be included in the jieba default dictionary. Jieba is able to identify new words, but you can add your own new words can ensure a higher accuracy.
- Usage： `jieba.load_userdict(file_name)` # file_name is a file-like object or the path of the custom dictionary
- The dictionary format is the same as that of `dict.txt` : one word per line; each line is divided into three parts separated by a space: word, word frequency, POS tag. If `file_name` is a path or a file opened in binary mode, the dictionary must be UTF-8 encoded.
- The word frequency and POS tag can be omitted respectively. The word frequency will be filled with a suitable value if omitted.

For example:

```
创新办 3 i
云计算 5
凯特琳 nz
台中
```

- Change a Tokenizer's `tmp_dir` and `cache_file` to specify the path of the cache file, for using on a restricted file system.
- Example:

```
云计算 5
李小福 2
创新办 3

[Before]： 李小福 / 是 / 创新 / 办 / 主任 / 也 / 是 / 云 / 计算 / 方面 / 的 / 专家 /

[After]：  李小福 / 是 / 创新办 / 主任 / 也 / 是 / 云计算 / 方面 / 的 / 专家 /
```

Modify dictionary

- Use `add_word(word, freq=None, tag=None)` and `del_word(word)` to modify the dictionary dynamically in programs.
- Use `suggest_freq(segment, tune=True)` to adjust the frequency of a single word so that it can (or cannot) be segmented.
- Note that HMM may affect the final result.

Example:

```
>>> print('/'.join(jieba.cut('如果放到post中将出错。', HMM=False)))
如果/放到/post/中将/出错/。
>>> jieba.suggest_freq(('中', '将'), True)
494
>>> print('/'.join(jieba.cut('如果放到post中将出错。', HMM=False)))
如果/放到/post/中/将/出错/。
>>> print('/'.join(jieba.cut('「台中」正确应该不会被切开', HMM=False)))
「/台/中/」/正确/应该/不会/被/切开
>>> jieba.suggest_freq('台中', True)
69
>>> print('/'.join(jieba.cut('「台中」正确应该不会被切开', HMM=False)))
「/台中/」/正确/应该/不会/被/切开
```

3. Keyword Extraction

```
import jieba.analyse
```

- `jieba.analyse.extract_tags(sentence, topK=20, withWeight=False, allowPOS=())`
  - `sentence` : the text to be extracted
  - `topK` : return how many keywords with the highest TF/IDF weights. The default value is 20
  - `withWeight` : whether return TF/IDF weights with the keywords. The default value is False
  - `allowPOS` : filter words with which POSs are included. Empty for no filtering.
- `jieba.analyse.TFIDF(idf_path=None)` creates a new TFIDF instance, `idf_path` specifies IDF file path.

Example (keyword extraction)

https://github.com/fxsjy/jieba/blob/master/test/extract_tags.py

Developers can specify their own custom IDF corpus in jieba keyword extraction

- Usage： `jieba.analyse.set_idf_path(file_name) # file_name is the path for the custom corpus`
- Custom Corpus Sample：https://github.com/fxsjy/jieba/blob/master/extra_dict/idf.txt.big
- Sample Code：https://github.com/fxsjy/jieba/blob/master/test/extract_tags_idfpath.py

Developers can specify their own custom stop words corpus in jieba keyword extraction

- Usage： `jieba.analyse.set_stop_words(file_name) # file_name is the path for the custom corpus`

- Custom Corpus Sample：https://github.com/fxsjy/jieba/blob/master/extra_dict/stop_words.txt
- Sample Code：https://github.com/fxsjy/jieba/blob/master/test/extract_tags_stop_words.py

There's also a TextRank implementation available.

Use: `jieba.analyse.textrank(sentence, topK=20, withWeight=False, allowPOS=('ns', 'n', 'vn', 'v'))`

Note that it filters POS by default.

`jieba.analyse.TextRank()` creates a new TextRank instance.

4. Part of Speech Tagging

- `jieba.posseg.POSTokenizer(tokenizer=None)` creates a new customized Tokenizer. `tokenizer` specifies the jieba.Tokenizer to internally use. `jieba.posseg.dt` is the default POSTokenizer.
- Tags the POS of each word after segmentation, using labels compatible with ictclas.
- Example:

```
>>> import jieba.posseg as pseg
>>> words = pseg.cut("我爱北京天安门")
>>> for w in words:
...    print('%s %s' % (w.word, w.flag))
...
我 r
爱 v
北京 ns
天安门 ns
```

5. Parallel Processing

- Principle: Split target text by line, assign the lines into multiple Python processes, and then merge the results, which is considerably faster.
- Based on the multiprocessing module of Python.
- Usage:
  - `jieba.enable_parallel(4)` # Enable parallel processing. The parameter is the number of processes.
  - `jieba.disable_parallel()` # Disable parallel processing.
- Example: https://github.com/fxsjy/jieba/blob/master/test/parallel/test_file.py
- Result: On a four-core 3.4GHz Linux machine, do accurate word segmentation on Complete Works of Jin Yong, and the speed reaches 1MB/s, which is 3.3 times faster than the single-process version.
- **Note** that parallel processing supports only default tokenizers, `jieba.dt` and `jieba.posseg.dt` .

6. Tokenize: return words with position

- The input must be unicode
- Default mode

```
result = jieba.tokenize(u'永和服装饰品有限公司')
for tk in result:
    print("word %s\t\t start: %d \t\t end:%d" % (tk[0],tk[1],tk[2]))
```

```
word 永和                start: 0                end:2
word 服装                start: 2                end:4
word 饰品                start: 4                end:6
word 有限公司            start: 6                end:10
```

- Search mode

```
result = jieba.tokenize(u'永和服装饰品有限公司',mode='search')
for tk in result:
    print("word %s\t\t start: %d \t\t end:%d" % (tk[0],tk[1],tk[2]))
```

```
word 永和                start: 0                end:2
word 服装                start: 2                end:4
word 饰品                start: 4                end:6
word 有限                start: 6                end:8
word 公司                start: 8                end:10
word 有限公司            start: 6                end:10
```

7. ChineseAnalyzer for Whoosh

- `from jieba.analyse import ChineseAnalyzer`
- Example: https://github.com/fxsjy/jieba/blob/master/test/test_whoosh.py

8. Command Line Interface

```
$> python -m jieba --help
Jieba command line interface.

positional arguments:
  filename              input file

optional arguments:
  -h, --help            show this help message and exit
  -d [DELIM], --delimiter [DELIM]
                        use DELIM instead of ' / ' for word delimiter; or a
                        space if it is used without DELIM
  -p [DELIM], --pos [DELIM]
                        enable POS tagging; if DELIM is specified, use DELIM
                        instead of '_' for POS delimiter
  -D DICT, --dict DICT  use DICT as dictionary
  -u USER_DICT, --user-dict USER_DICT
                        use USER_DICT together with the default dictionary or
                        DICT (if specified)
  -a, --cut-all         full pattern cutting (ignored with POS tagging)
  -n, --no-hmm          don't use the Hidden Markov Model
  -q, --quiet           don't print loading messages to stderr
  -V, --version         show program's version number and exit

If no filename specified, use STDIN instead.
```

Initialization

By default, Jieba don't build the prefix dictionary unless it's necessary. This takes 1-3 seconds, after which it is not initialized again. If you want to initialize Jieba manually, you can call:

```
import jieba
jieba.initialize()  # (optional)
```

You can also specify the dictionary (not supported before version 0.28) :

```
jieba.set_dictionary('data/dict.txt.big')
```

Using Other Dictionaries

It is possible to use your own dictionary with Jieba, and there are also two dictionaries ready for download:

1. A smaller dictionary for a smaller memory footprint: https://github.com/fxsjy/jieba/raw/master/extra_dict/dict.txt.small

2. There is also a bigger dictionary that has better support for traditional Chinese (繁體): https://github.com/fxsjy/jieba/raw/master/extra_dict/dict.txt.big

By default, an in-between dictionary is used, called `dict.txt` and included in the distribution.

In either case, download the file you want, and then call `jieba.set_dictionary('data/dict.txt.big')` or just replace the existing `dict.txt` .

Segmentation speed

- 1.5 MB / Second in Full Mode
- 400 KB / Second in Default Mode
- Test Env: Intel(R) Core(TM) i7-2600 CPU @ 3.4GHz；《围城》.txt