

## ● 特点

### 1. 三种分词模式：

- 1) 精确模式，试图将句子最精确地切开，适合文本分析；
- 2) 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；
- 3) 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。

### 2. 支持繁体分词

### 3. 支持自定义词典

### 4. MIT 授权协议

## ● 主要功能

### 1. 分词

```
# 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))
```

Building prefix dict from the default dictionary ...  
Dumping model to file cache C:\Users\think\AppData\Local\Temp\jieba.cache  
Loading model cost 4.455 seconds.  
Prefix dict has been built successfully.

Full Mode: 我/ 来到/ 北京/ 清华/ 清华大学/ 华大/ 大学  
Default Mode: 我/ 来到/ 北京/ 清华大学  
他, 来到, 了, 网易, 杭研, 大厦  
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, 日本京都大学, 深造

- 1) jieba.cut 3 个参数：字符串、cut\_all 参数、是否使用 [HMM 模型](#)

不定义 cut\_all 参数时，默认是精确模式

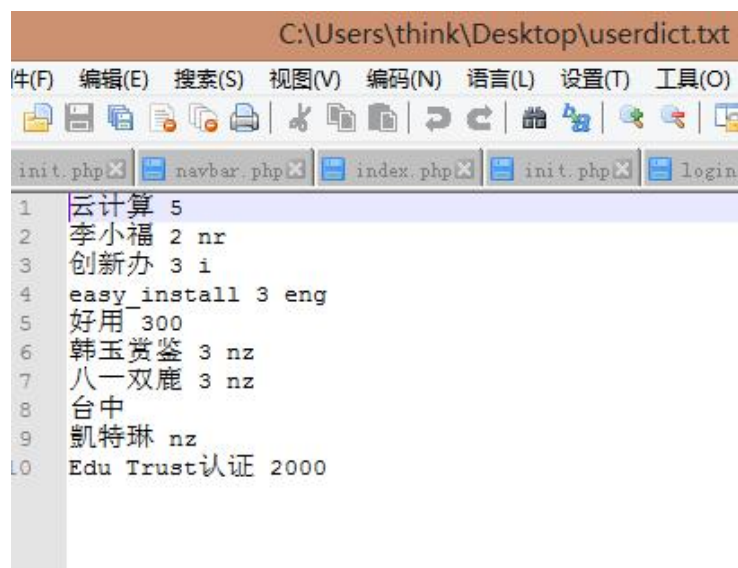
- 2) jieba.cut\_for\_search 方法两个参数：字符串； HMM 模型
- 3) 字符串要求：**unicode** 或 **UTF-8** 字符串、**GBK** 字符串
- 4) [生成器](#)，可用 for 循环获得词

## 2. 自定义词典

可以自定义词典，方便个性化需求与纠错

- 1) 创建词典 txt

词（空格）词频（空格）词性



```
C:\Users\think\Desktop\userdict.txt
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O)
init.php navbar.php index.php init.php login
1 云计算 5
2 李小福 2 nr
3 创新办 3 i
4 easy_install 3 eng
5 好用 300
6 韩玉赏鉴 3 nz
7 八一双鹿 3 nz
8 台中
9 凯特琳 nz
10 Edu Trust认证 2000
```

## 2) 调整

A. 使用 `add_word(word, freq=None, tag=None)` 和 `del_word(word)`

可在程序中动态修改词典。

word 词, freq 词频, tag 词性

```
from __future__ import print_function, unicode_literals
import sys
sys.path.append("../")
import jieba
jieba.load_userdict(r'C:\\Users\\think\\Desktop\\userdict.txt')
import jieba.posseg as pseg

jieba.add_word('石墨烯') #添加自定义词
jieba.add_word('凯特琳')
jieba.del_word('自定义词') #删除自定义词

test_sent = (
    "李小福是创新办主任也是云计算方面的专家；什么是八一双鹿\n"
    "例如我输入一个带“韩玉赏鉴”的标题，在自定义词库中也增加了此词为N类\n"
    "「台中」正确应该不会被切開。mac上可分出「石墨烯」；此時又可以分出來凱特琳了。"
)
words = jieba.cut(test_sent)
print(' '.join(words))

print("\n"*5)

result = pseg.cut(test_sent)

for w in result:
    print(w.word, "/", w.flag, ", ", end=' ')
```

李小福/是/创新办/主任/也/是/云计算/方面/的/专家/;/ /什么/是/八一双鹿/  
/例如/我/输入/一个/带/"韩玉赏鉴"/的/标题/, /在/自定义/词库/中/也/增加/了/此/词为/N类/  
/「台中」/正确/应该/不/被/切開/。/mac/上/可/分出/"石墨烯"/;/ /此時/又/可以/分出/來/凱特琳/了/。

- B. 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。

```
import jieba
words =jieba.cut("我们中出了一个叛徒", HMM=False)
jieba.suggest_freq('中出'), True)
print ('/'.join(words))
```

我们/中出/了/一个/叛徒

- C. jieba.posseg 词性标注    `posseg.cut()`

word 词语 flag 词性

李小福 /nr, 是 /v, 创新办 /i, 主任 /b, 也 /d, 是 /v, 云计算 /x, 方面 /n, 的 /uj, 专家 /n, ; /x, /x, 什  
么 /r, 是 /v, 八一双鹿 /nz,  
/x, 例如 /v, 找 /r, 输入 /v, 一个 /n, 带 /v, “ /x, 韩玉赏鉴 /nz, ” /x, 的 /uj, 标题 /n, , /x, 在 /p  
, 自定义 /l, 词库 /n, 中 /f, 也 /d, 增加 /v, 了 /ul, 此 /r, 词 /n, 为 /p, N /eng, 类 /q,  
/x, 「 /x, 台中 /s, 」 /x, 正确 /ad, 應該 /v, 不 /d, 會 /v, 被 /p, 切開 /ad, 。 /x, mac /eng, 上 /f,  
可 /v, 分出 /v, 「 /x, 石墨烯 /x, 」 /x, ; /x, 此時 /c, 又 /d, 可以 /c, 分出 /v, 來 /zg, 凱特琳 /nz, 了  
/ul, 。 /x,

### 3. 基于 TF-IDF 算法的关键词抽取

一个容易想到的思路, 就是找到出现次数最多的词。如果某个词很重要, 它应该在这篇文章中多次出现。于是, 我们进行“词频” (Term Frequency, 缩写为 TF) 统计。

用统计学语言表达, 就是在词频的基础上, 要对每个词分配一个重要性权重。最常见的词 (“的”、“是”、“在”) 给予最小的权重, 较常见的词 (“中国”) 给予较小的权重, 较少见的词 (“蜜蜂”、“养殖”) 给予较大的权重。这个权重叫做“逆文档频率” (Inverse Document Frequency, 缩写为 IDF), 它的大小与一个词的常见程度成反比。

将这两个值相乘, 就得到了一个词的 TF-IDF 值。某个词对文章的重要性越

高，它的 TF-IDF 值就越大。所以，排在最前面的几个词，就是关键词。

```
print("---案例1---"*3)
txt='有的人活着 他已经死了； 有的人死了 他还活着。 有的人 骑在人民头上：“呵，我多伟大！”
Key=jieba.analyse.extract_tags(txt, topK=3)
print(Key)
#-----
print("---案例2---"*3)
# 字符串前面加u表示使用unicode编码
content = u'中国特色社会主义是我们党领导的伟大事业，全面推进党的建设新的伟大工程，是这一伟

keywords = jieba.analyse.extract_tags(content, topK=5, withWeight=True, allowPOS=())
# 访问提取结果
for item in keywords:
    # 分别为关键词和相应的权重
    print(item[0], item[1])
```

---案例1-----案例1-----案例1---  
['活着', '伟大', '头上']  
---案例2-----案例2-----案例2---  
党的建设 0.47331204260459014  
管党 0.3919595902590164  
伟大工程 0.3771404058754098  
伟大事业 0.3669713918327869  
才能 0.26339384065180327

# 第一个参数：待提取关键词的文本

# 第二个参数 topK：返回关键词的数量，重要性从高到低排序

# 第三个参数 withWeight：是否同时返回每个关键词的权重

# 第四个参数 allowPOS：词性过滤，为空表示不过滤，若提供则仅返回符合词性要求的关键词

```
1 import sys
2 sys.path.append('../')
3
4 import jieba
5 import jieba.analyse
6 from optparse import OptionParser
```

<https://www.cnblogs.com/paulwhw/p/9065074.html>