# 《Jieba源码研读笔记》—Version 1.0

## 基本功能介绍

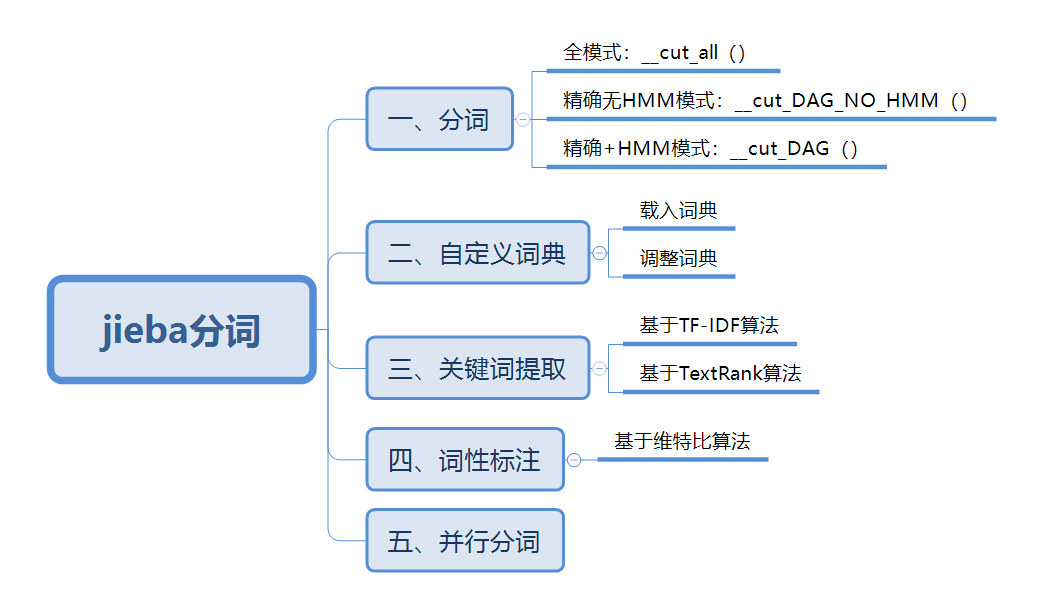


图1-1 Jieba分词功能



图1-2 Jieba分词包功能图

### 1.1 分词模式

#### 1.1.1 模式划分

* **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

#### 1.1.2 安装说明

* **全自动安装**：easy\_install jieba 或者 pip install jieba / pip3 install jieba，在anaconda下也可以使用conda 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

#### 1.1.3 分词算法

* 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

基本概念：

DAG：[百度百科](https://baike.baidu.com/item/%E6%9C%89%E5%90%91%E6%97%A0%E7%8E%AF%E5%9B%BE/10972513?fr=aladdin&fromtitle=DAG&fromid=17610231)、[简书](https://www.jianshu.com/p/c7dbc61cda1d)

动态规划：[简书](https://www.jianshu.com/p/8bfa28cc3893)

Viterbi算法：[简书](https://www.jianshu.com/p/6b2794f7a7ca)、[知乎](https://www.zhihu.com/question/20136144)

#### 1.1.4 分词算法流程

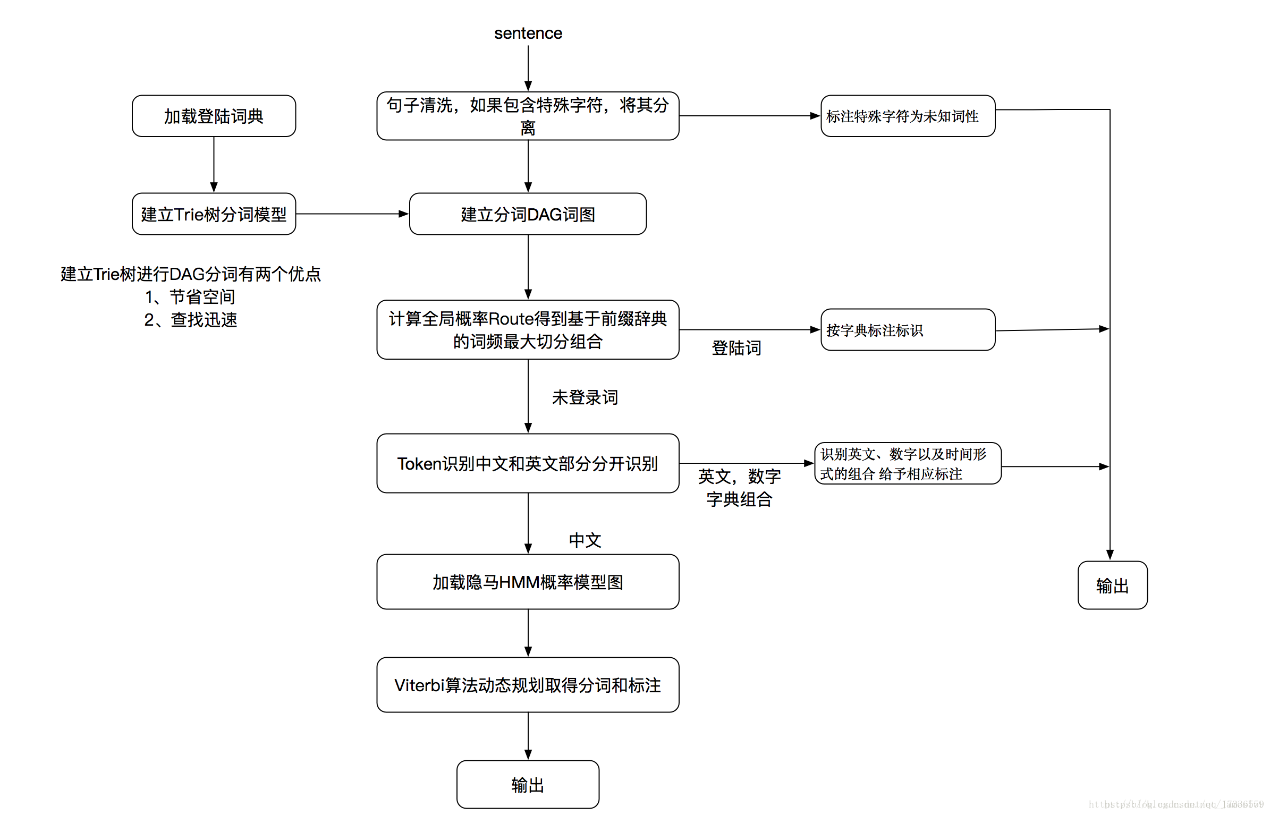


图1-3 Jieba分词技术路线图

### 自定义词典

#### 1.2.1 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

**示例1：** [**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)

#### 1.2.2 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

### 关键词提取

#### 1.3.1 基于 TF-IDF 算法的关键词抽取

* 利用 import Jieba.analyse 导入TF-IDF包
* jieba.analyse.extract\_tags(sentence, topK=20, withWeight=False, allowPOS=())
  + 1. sentence 为待提取的文本；
    2. topK 为返回几个 TF/IDF 权重最大的关键词，默认值为 20
    3. withWeight 为是否一并返回关键词权重值，默认值为 False
    4. 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

**算法原理：**[**百度百科**](https://baike.baidu.com/item/tf-idf/8816134?fr=aladdin)**、**[**CSDN**](https://blog.csdn.net/qq_20989105/article/details/82685162)

**示例2**：**[TF-IDF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）**](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）](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](https://github.com/fxsjy/jieba/blob/master/test/extract_tags_with_weight.py)**

#### 1.3.2 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
* 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

**算法原理：** [**TextRank论文**](http://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf)

**示例3 ：[TEXT-RANK关键词抽取](https://github.com/fxsjy/jieba/blob/master/test/demo.py)**

### 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。

具体内容：见P

### 并行分词

#### 1.5.1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

#### 1.5.2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

**示例4：[并行分词](https://github.com/fxsjy/jieba/blob/master/test/parallel/test_file.py)**

* 注意：并行分词仅支持默认分词器 jieba.dt 和 jieba.posseg.dt。

## 分词部分源码简要介绍

### 2.1 Jieba包分词功能分析

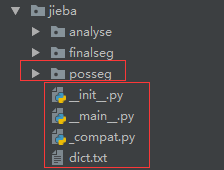


图2-1 结巴分词包结构

Jieba 的分词功能是由**Jieba模块本身以及finalseg模块**来完成的。

而Jieba这个模块里包含了**\_\_init\_\_.py，\_\_main\_\_.py，\_compat.py**及**dict.txt**四个包：

1. **\_\_init\_\_.py**定义了Tokenizer类别及一些全局函数，用于分词本身；
2. **\_\_main\_\_.py**定义了Jieba在命令行里的使用方式；
3. **\_compat.py**用于处理Python2/3相容性的问题；
4. **dict.txt**则是字典，记录了各词的词频及词性。

主要介绍：**\_\_init\_\_.py、\_compat.py**

### 2.2 Jieba/****init****.py

#### 2.2.1 Tokenizer类

在\_\_init\_\_.py这个python文件里面，定义了一个叫做Tokenizer（）的类。它拥有cut（），cut\_for\_search（），tokenize（）等多种方法，负责了分词的工作。

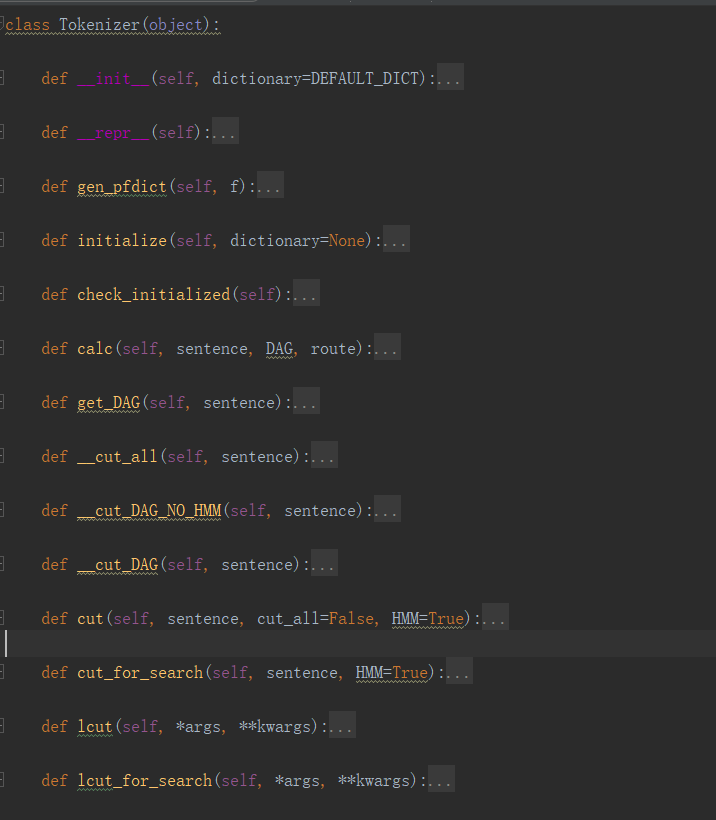


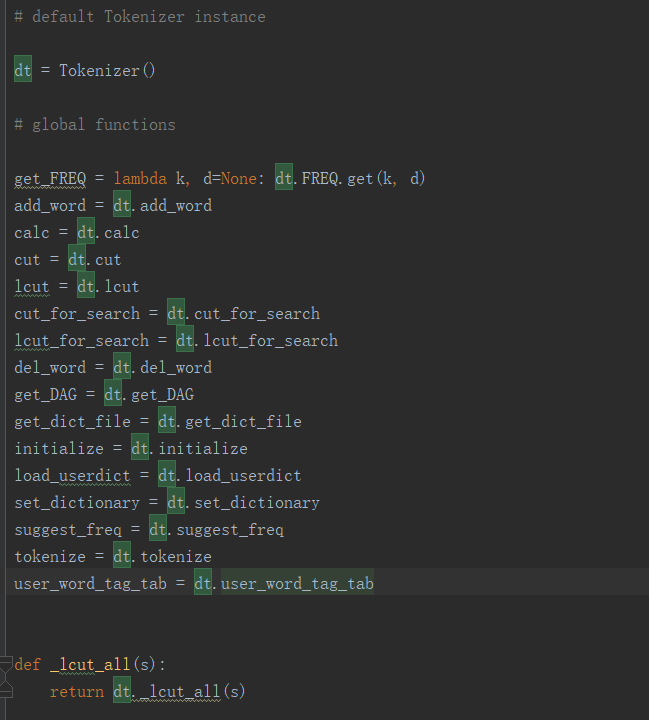
图2-2（a） Tokenizer 函数



图2-2（b） Tokenizer 函数

#### 2.2.2 Tokenizer全局函数

在定义好Tokenizer类别后，\_\_init\_\_.py里建立了一个Tokenizer类别的dt对象。然后逐一定义全局函数，并将它们指向dt中相对应的函数，如图2-3（a）、(b)所示：



2-3（a） Tokenizer全局函数



2-3（b） Tokenizer全局函数

**问题一：**为什么可以直接调用jieba.cut来分词，这是怎么做到的呢？

解析：由于存在有cut = dt.cut这一句话，它定义了一个全局函数cut，并把它指向dt对象的cut函数。如此一来，就可以不用自己新建一个Tokenizer对象，而是可以直接使用jieba.cut来分词。

**上述全局函数皆指向dt所拥有的函数。除了这些函数外，还定义一个全局函数\_get\_abs\_path、\_replace\_file。**

#### 2.2.3 全局函数\_get\_abs\_path

全局函数\_get\_abs\_path如下图2-4所示：

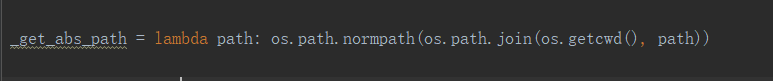


图2-4 \_get\_abs\_path函数

这个函数的参数path是字典的名称，它的作用是在字典名称前加上当前路径，然后把路径正规化后回传。

#### 2.2.4 全局函数\_replace\_file

全局函数\_replace\_file如下图2-5所示：

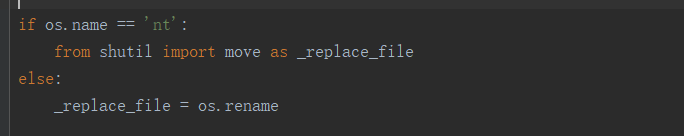


图2-5 \_replace\_file函数

这里使用if-else的写法是为了处理重命名函数在不同作业系统上的相容性，确保\_replace\_file在不同的作业系统上皆能运作。

参考：[I don’t understand os.name==‘nt’: . what is nt and os.name](https://teamtreehouse.com/community/i-dont-understand-osnament-what-is-nt-and-osname)，代码第一行的os.name == ''nt''代表当前的作业系统是Windows。而代码第一行os.name=='posix'则代表linux。

#### 2.2.5 正则表达式

这里定义了数个正则表达式，它们会在分词及载入字典时发挥作用。

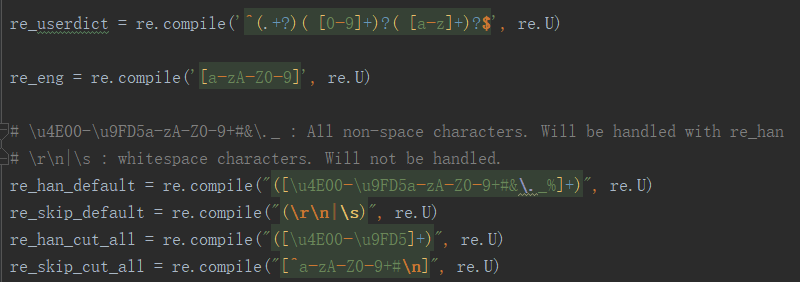


图2-6 jieba正则表达式

#### 2.2.6 log相关函数

default\_logger如字面上的意思，是这个脚本档中预设的logger。

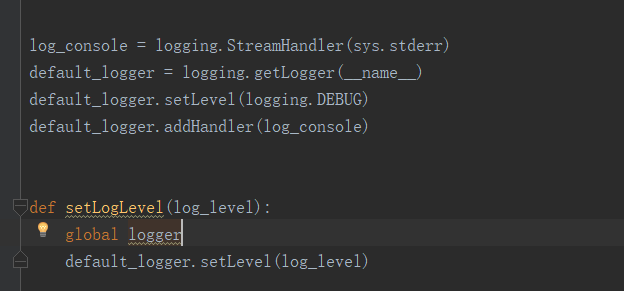


图2-7 Log相关函数

#### 2.2.7 并行分词相关函数

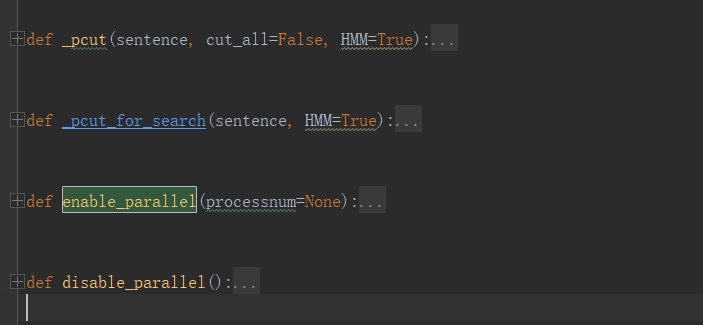


图2-8 并行分词相关函数

### 2.3 Jieba/****\_compat.py****

\_compat.py这个python文件，它被用于处理Python2/3之间相容的问题。

这个文件中定义了get\_module\_res、strdecode、resolve\_filename等读文件时会用到的函数，它们会在\_\_init\_\_.py中频繁地被调用。

#### 2.3.1 get\_module\_res函数

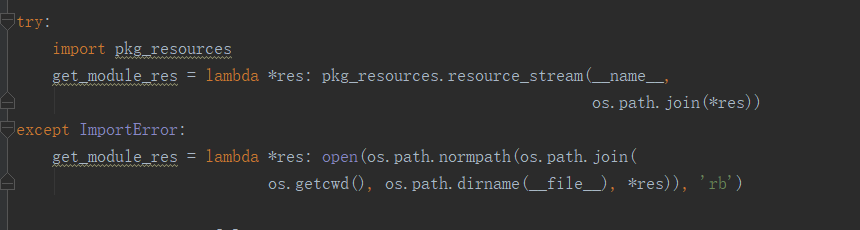


图2-9 get\_module\_res函数

#### 2.3.2 统一Python2/3函数名称

这里首先判断Python版本是否为Python2，并将它存到PY2这个变数里。接着是依据Python2/3的特性，一一定义text\_type及stringe\_types等。在Python3中，xrange变成range，遍历字典的方式也跟Python2有所不同。这里建立了数个函数，并将它们指向Python3中相对应功能的函数。这样一来，就可以统一以Python2的方式来呼叫他们。

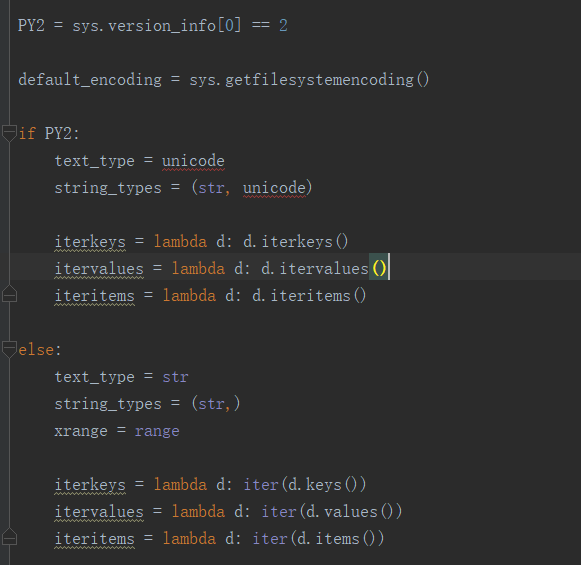


图 2-10 统一Python2/3函数名称

#### 2.3.3 strdecode函數

在使用Python 3的情况下，如果传入的sentence不是字串类型，而是bytes型别，就将它以utf-8编码转换成字串。如果解码失败，则改以gbk编码(简体中文)来转换。所以这个函数的作用就是确保sentence是字符串类型后回传。

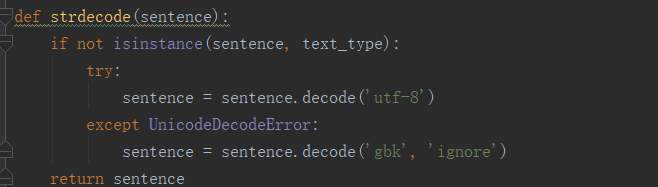
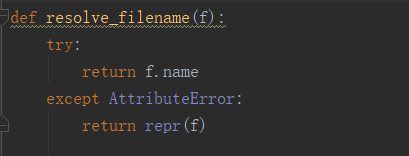


图 2-11 strdecode函数

#### 2.3.4 resolve\_filename函數

使用f = open(''xxx.txt'', ''r'')会得到一个\_io.TextIOWrapper型别的对象f。

resolve\_filename接受一个\_io.TextIOWrapper型别的对象当作参数，获取它的文件文件名后回传。



2-12 resolve\_filename函數

### 2.4 Jieba/****\_main\_.py****

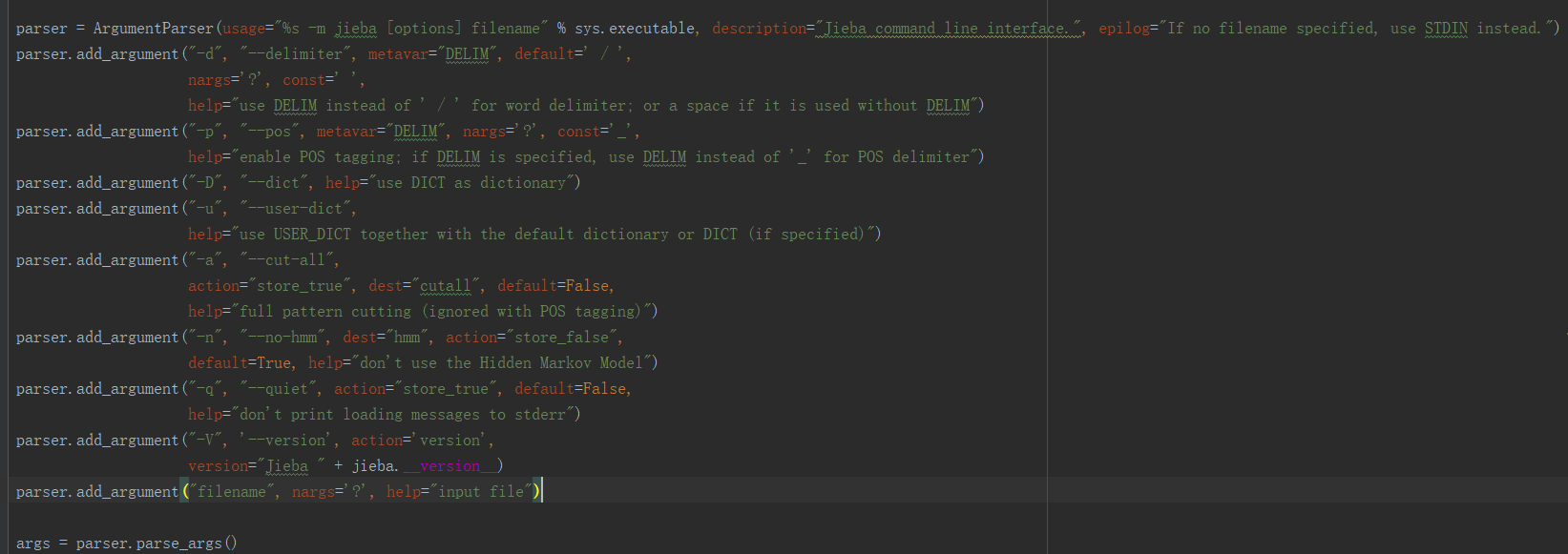
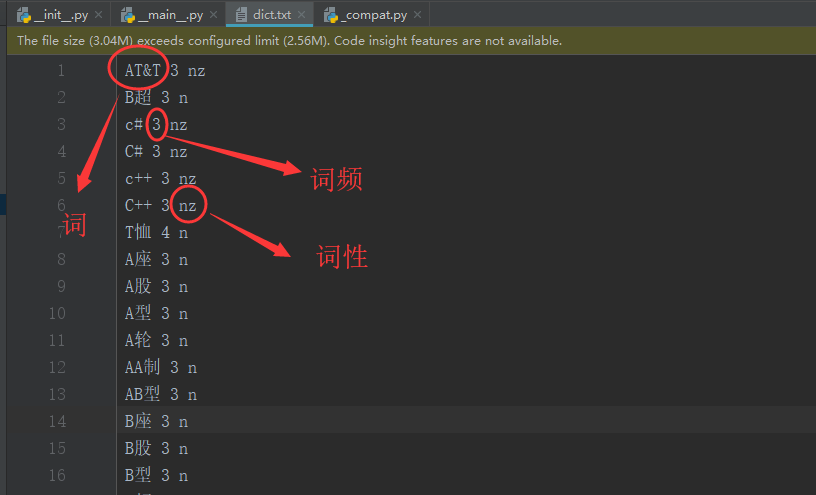


图2-13 Jieba命令行使用方式

### 2.5 Jieba/****dict.txt****

dict.txt 文件词[来源](https://github.com/fxsjy/jieba/issues/7)：





2-14 dict.txt文件

## 分词核心部分详解

Jieba的jieba/\_\_init\_\_.py是核心部分，通过\_\_init\_\_.py进入到分词的三种模式，而\_\_init\_\_.py的核心部份是Tokenizer类别，这将是本篇介绍的重点。

### 3.1 Tokenizer类各函数详解

#### 3.1.1 \_\_init\_\_函数

在Tokenizer类别中有\_\_init\_\_及initialize这两个函数，他们发挥的都是初始化的作用。

但是\_\_init\_\_函数是比较轻量级的，在该函数中只简单地定义了几个属性。分词所必需的字典载入则延后至initialize函数中完成。

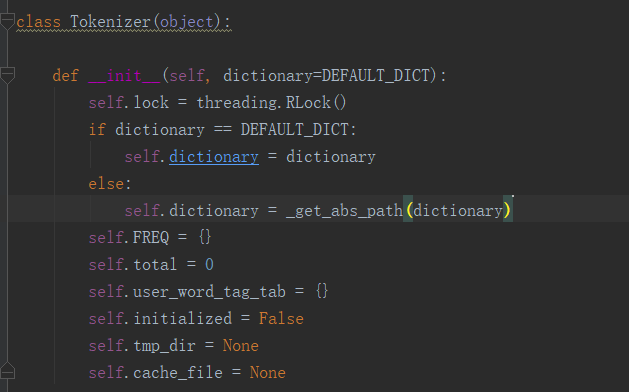


图3-1 \_init\_函数

#### 3.1.2 \_\_repr\_\_函数

这里覆写了object类别的\_\_repr\_\_函数

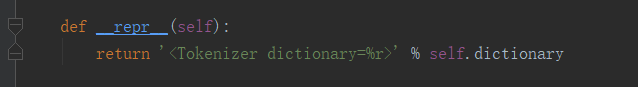


图3-2 \_\_repr\_\_函数

#### 3.1.3 get\_dict\_file函数

这个函数的作用是读取字典档案，开启后回传。它预设读取dict.txt，但使用者也可以自定义字典。

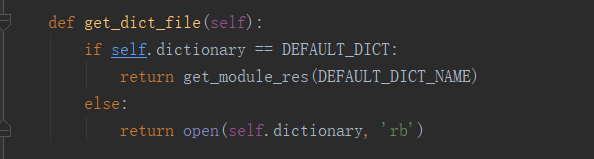
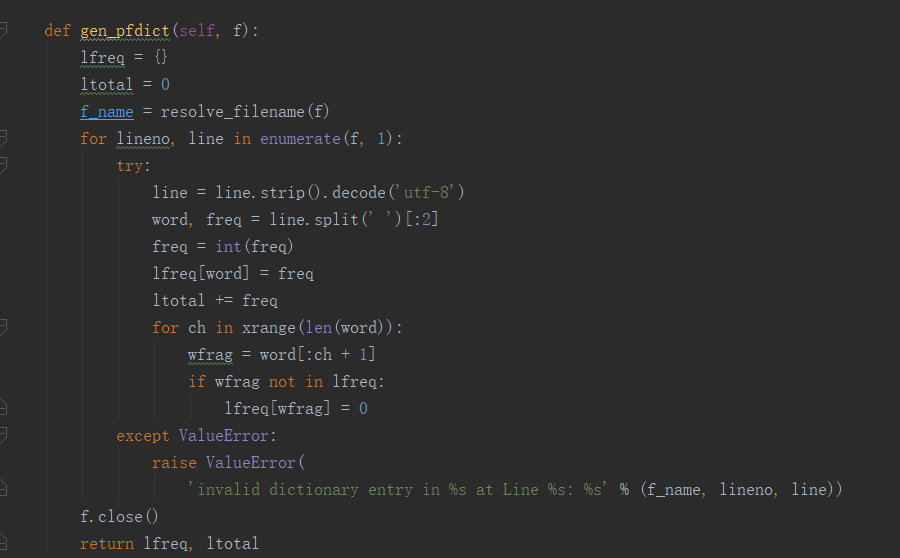


图3-3 get\_dict\_file函数

#### 3.1.4 gen\_pfdict函数

gen\_pfdict的作用是从一个己开启的字典file object中，获取每个词的出现频率以及所有词的出现次数总和。

以下是它的函数定义：



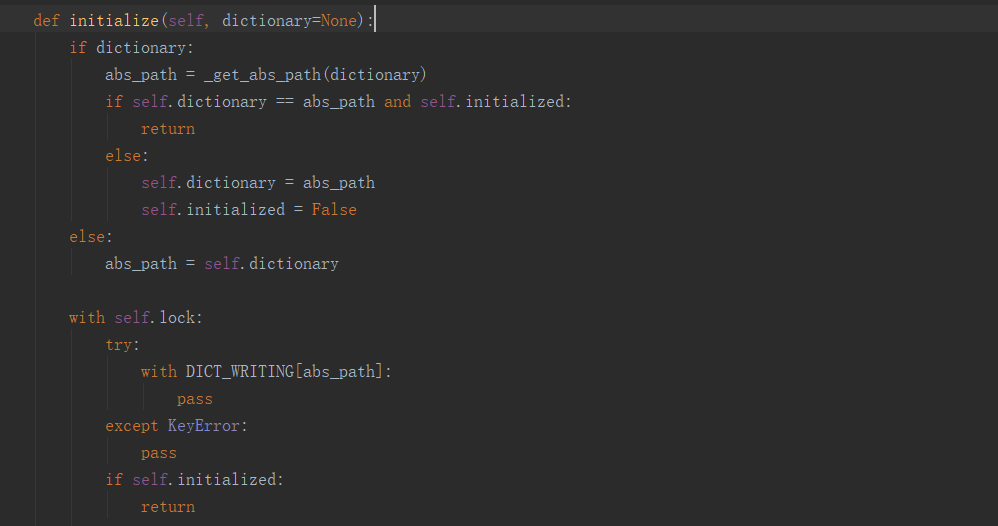
1. lfreq = {} #记录每个词的出现次数
2. ltotal = 0 #所有词出现次数的总和

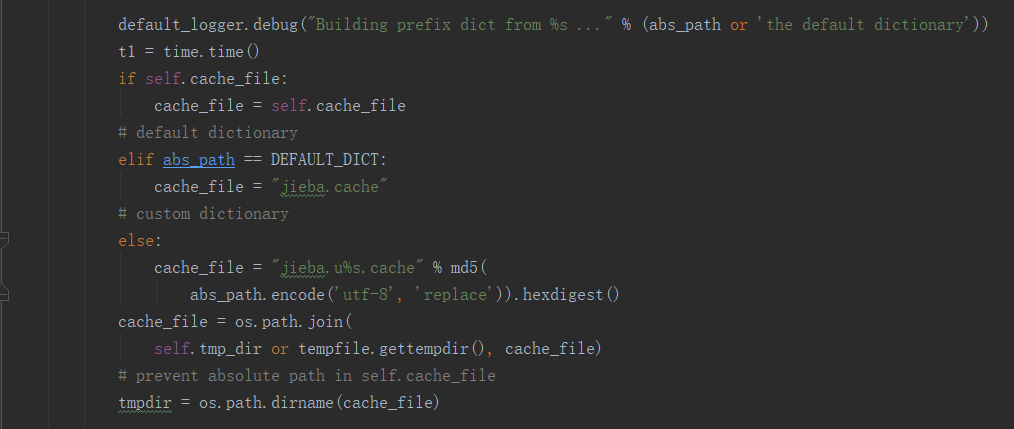
#### 3.1.5 initialize函数

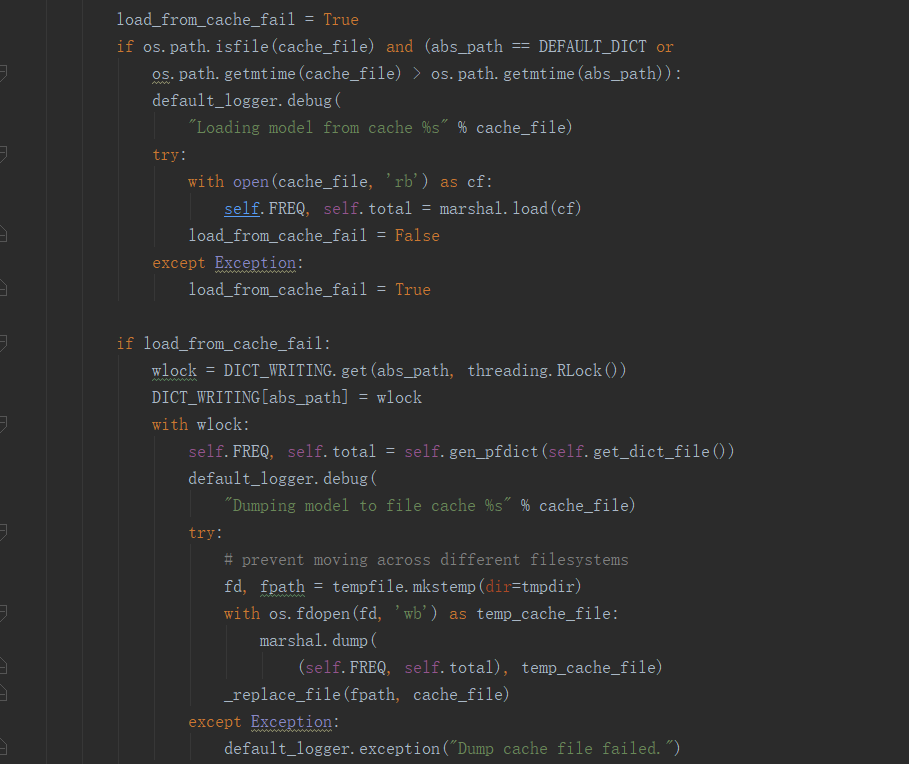
initialize函数的功能是载入字典，虽然与\_\_init\_\_函数一样都是用于初始化。但是它不像\_\_init\_\_是在物件创造时就被执行，而是在使用者要存取字典或是开始进行分词的时候才会执行。

initialize函数会调用前面介绍的get\_dict\_file，gen\_pfdict，\_get\_abs\_path，DICT\_WRITING，default\_logger等函数及变量。

在initialize函数的定义中，使用到了tempfile，marshal套件以及threading中的RLock类别。我们先来看看他们各自的功能：







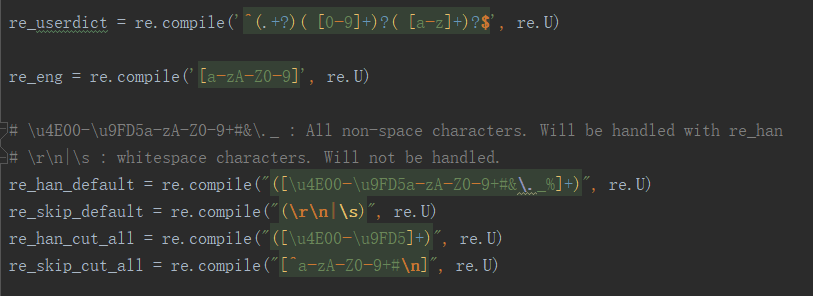
### 3.2 正则表达式

jieba包含的三大功能：分词、词性标注及关键词提取都需要用到正则表达式。

笔者将jieba/\_\_init\_\_.py、jieba/finalseg/\_\_init\_\_.py及jieba/posseg/\_\_init\_\_.py三个文件里出现的正则表达式集中在本节中介绍。

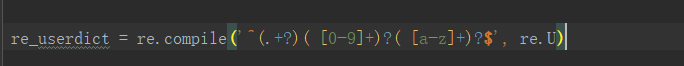
正则表达式学习链接：[学习地址1](https://www.tutorialspoint.com/python/python_reg_expressions.htm) 、[学习地址2](https://www.runoob.com/regexp/regexp-syntax.html)

#### 3.2.1 jieba/\_\_init\_\_.py 正则



包含有：re\_userdict 、re\_eng、re\_han\_default、re\_skip\_default、re\_han\_cut\_all、re\_skip\_cut\_all

1. re\_userdict是在解析dict.txt的时候使用的：



re.compile会将pattern编译成正则表达式物件。我们之后就可以用它来match或search其它串。

* 锚点 ^ 和 $ :^表示匹配字串的开头，$表示匹配字串的结尾
* 分组和捕获():()会创造一个捕获性分组，之后可以用re\_userdict.match(line).groups()来得到己匹配的分组
* lazy匹配，.+?：.+会贪心地(尽可能多地)匹配，加上?之后变成lazy匹配。如果实际去测试移除?之后的效果，会发现字串全部被分到第一个组别。
* 或运算符[]：如[0-9]匹配0到9中的其中一个字

#### 3.2.2 jieba/finalseg/\_\_init\_\_.py 正则

#### 3.2.3 jieba/posseg/\_\_init\_\_.py 正则