1 实验基本信息

实验名称：对分词工具 thulac 和 jieba 进行探究以及其在词云上的实现

引用网址：https://github.com/fxsjy/jieba https://github.com/thunlp/THULAC-Python

引用（THULAC）： Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, Zhiyuan Liu. THULAC: An Efficient Lexical Analyzer for Chinese. 2016.

实验设备：windos10 计算机、VScode 软件

实验时间：2020 年 5 月

2 实验简介

分词是自然语言的基础，文章内容主要介绍的是由清华大学自然语言处理与社会人文计算实验室研制出的一套中文分析工具包-THULAC(THU Lexical Analyzer for Chinese), 其具有中文分词和词性标注的功能，本实验在对其进行验证使用之前，首先验证使用了目前当下最流行的中分分词工具---jieba，然后通过 VScode 探索 THULAC 的 Python 版的基本功能，并对其进行分析，并且在此基础上，还会简要介绍词云，并将 THULAC 在词云上实现了应用。

3 研究意义

词是最小的能够独立运用的语言单位，而很多孤立语和黏着语也称亚系语言(如汉语、日语、越南语、藏语等)的文本的词与词之间没有任何空格之类的显示标志指示词的边界，因此中文自然语言处理的基础就是中文分词，没有中文分词就很难将中文语言量化。本实验中研究的就是目前应用最为广泛的中文分词工具 jieba 以及由清华大学开发的 THULAC,并实践其基本功能以及在词云上的应用。

4 实验过程：

4.1 jieba 分词分析：

4.1.1 jieba 工作原理：

(1). jieba 分词采用了基于 Trie 树结构的算法，jieba 分词利用该算法高效实现了词图扫描，并且利用词图扫描将得到句子中汉字所有的成词可能，并且将这些所有成词可能的情况构成有向无环图（DAG）为下一步打下基础。 通过分词 jieba 分词的源码可以发现，jieba 分词本身就包含了一个有 2 万多词条的词典。基于 Trie 结构的扫描就是将这些词条放到 Trie 的树结构之中，一旦扫描的词条中和这些放在 Trie 结构中的词条有着相同的前缀，那么就实现了快速查找。

(2). jieba 分词中最大概率路径的实现采用了动态规划查找的方法，动态规划查找可以找出根据词频的最大切分组合。分析 jieba 分词的源码可以发现 jieba 分词不仅仅将字典生成 Trie 树，同时，将把每个词的出现次数转换为了频率。

(3). 对于在 jieba 分词自带的词典中未出现的词，jieba 分词采用了 Viterbi 算法，并且将用于汉字成词的 HMM 模型应用其中。

4. 1. 2 实验步骤及分析：

（1）安装：使用 pip install 进行安装，即输入命令：

```
pip install jieba
```

（2）jieba 分词支持三种模式，即精确模式、全模式和搜索引擎模式。我们分别对其进行了测试。

```
import jieba
seg_list = jieba.cut("前年我来到上海复旦大学", cut_all=True)
print("Full Mode: " + "/ ".join(seg_list))  # 全模式
seg_list = jieba.cut("前年我来到上海复旦大学", cut_all=False)
print("Default Mode: " + "/ ".join(seg_list))  # 精确模式
seg_list = jieba.cut("前年我来到上海复旦大学")  # 默认是精确模式
print(", ".join(seg_list))
seg_list = jieba.cut_for_search("小王本科毕业于北京交通大学，将要工作于字节跳动公司")  # 搜索引擎模式
print(", ".join(seg_list))
```

其运行结果为：

```
Prefix dict has been built successfully.
Full Mode: 前年/ 我/ 来到/ 上海/ 上海复旦大学/ 复旦/ 复旦大学/ 大学
Default Mode: 前年/ 我/ 来到/ 上海复旦大学
前年, 我, 来到, 上海复旦大学
小王, 本科, 毕业, 本科毕业, 于, 北京, 交通, 大学, , , 将要, 工作, 于, 字节, 跳动, 公司
PS C:\Users\dell\Desktop\大作业>
```

我们可以看到，对与精确模式下，其试图将句子最精确地切开，适合文本分析而全模式是将句子中所有的可以成词的词语都扫描出来，如“上海复旦大学”在精确模式为“上海/ 上海复旦大学/ 复旦/ 复旦大学/ 大学”，而在全模式下只是为“上海复旦大学”。对于搜索引擎模式。其是利用 jieba.cut_for_search 实现，我们可以看到例子中的词都被划分为比较短的词他是在在精确模式的基础上，对长词再次切分，提高召回率，从而适合用于搜索引擎分词。

（3）词性标注：jieba 有一个很重要的功能就是词性标注，即对句子中每个词的词性进行标注，其部分词性标签如下：

标签	含义	标签	含义	标签	含义	标签	含义
n	普通名词	f	方位名词	s	处所名词	t	时间
nr	人名	ns	地名	nt	机构名	nw	作品名
nz	其他专名	v	普通动词	vd	动副词	vn	名动词
a	形容词	ad	副形词	an	名形词	d	副词
m	数量词	q	量词	r	代词	p	介词
c	连词	u	助词	xc	其他虚词	w	标点符号
PER	人名	LOC	地名	ORG	机构名	TIME	时间

我们对其进行功能测试如下：

```
 import jieba.posseg as pseg
words = pseg.cut("我深深地爱着我的祖国")
# words 类别为: generator
for word, flag in words:
 print('%s %s' % (word, flag))
```

其运行结果为：

```
Prefix dict has been built successfully.
我 r
深深地 i
爱 v
着 uz
我 r
的 uj
祖国 n
```

可以看到机器将“我”认作为代词，“着”“的”为固定助词，“深深地”认作为副词，“祖国”为名词，能够比较准确的对其词性进行标注。

（3）关键词提取：jieba 的关键词提取主要有两种，一种是基于 TF-IDF 算法，一种是基于 TextRank 算法，在这里我们主要介绍基于 TF-IDF 算法的关键词抽取。

原理：TF 即词频，它定义为 TF=(某个词在文章中的出现次数)/文章的总词数。而当如果某个词比较少见，但是它在这篇文章中多次出现，那么它很可能就反映了这篇文章的特性，正是我们所需要的关键词。而 IDF（逆文档频率）就是描述这种关系，它定义为：

$$IDF = \log(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数+1}})$$

而 TF-IDF 就是将 TF 和 IDF 结合起来的一种算法，即与一个词在文档中的出现次数成正比，与该词在整个语言中的出现次数成反比。其代码格式如下：

```
jieba.analyse.extract_tags(sentence, topK=20, withWeight=False, allowPOS=())
```

其中 sentence 为待提取的文本；topK 为返回几个 TF/IDF 权重最大的 g 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。我们选用人民日报的一篇抗疫文章其进行测试，其代码和结果如下：

```
import jieba
import jieba.analyse as analyse
lines=open('RMRB.txt',encoding='utf-8').read()
print ("  ".join(analyse.extract_tags(lines, topK=20, withWeight=False,
 allowPOS=())))
```

运行结果：

```
Prefix dict has been built successfully.
患者 救治 重症 生命 疫情 抗疫 武汉 医护人员 至上 挽救 确诊 病例 清零 新冠 他活 泪点 25 ECMO 50 四省
PS C:\Users\dell\Desktop\大作业>
```

我对其文章进行阅读后，发现其结构基本符合其关键词的标准。

4.2 THULAC 分词分析:

下面我们对清华的 THULAC 进行分析。首先其原理我们经过分析发现应该是基于概率语言模型的方法，其有一下特点：

a.能力强。利用我们集成的目前世界上规模最大的人工分词和词性标注中文语料库（约含 5800 万字）训练而成，模型标注能力强大。

b.准确率高。该工具包在标准数据集 Chinese Treebank（CTB5）上分词的 F1 值可达 97.3%，词性标注的 F1 值可达到 92.9%，与该数据集上最好方法效果相当。

c.速度较快。同时进行分词和词性标注速度为 300KB/s，每秒可处理约 15 万字。只进行分词速度可达到 1.3MB/s

其实验内容如下：

（1）安装。其安装步骤与 jieba 相同，都是使用 pip install 命令，即

```
    pip install thulac
```

（2）分词及词性标注：THULAC 中常用的词性标注符号如下：

```
n/名词 np/人名 ns/地名 ni/机构名 nz/其它专名
m/数词 q/量词 mq/数量词 t/时间词 f/方位词 s/处所词
v/动词 a/形容词 d/副词 h/前接成分 k/后接成分
i/习语 j/简称 r/代词 c/连词 p/介词 u/助词 y/语气助词
e/叹词 o/拟声词 g/语素 w/标点 x/其它
```

其词性标注的代码如下：

```
import thulac
thu1 = thulac.thulac()  #默认模式
text = thu1.cut("我深深地爱着我的祖国", text=True)  #进行一句话分词
print(text)
```

其运行结果为：

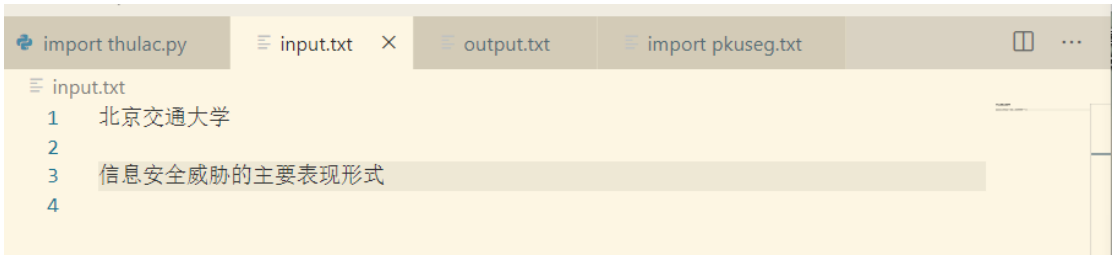
```
Model loaded succeed
我_r 深深_d 地_u 爱_v 着_u 我_r 的_u 祖国_n
PS C:\Users\dell\Desktop\大作业>
```

我们可以发现，其词性标注与 jieba 有所不同，相对于 jieba 的分词，THULAC 更加详细，比如 jieba 里的“深深地”在这里被分成了“深深”与“地”。

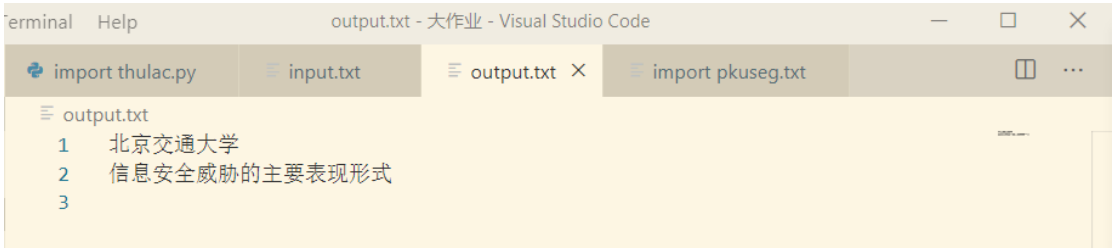
（3）去除空格：

```
import thulac
with open('input.txt','r',encoding = 'utf-
8') as fr,open('output.txt','w',encoding = 'utf-8') as fd:
        for text in fr.readlines():
                if text.split():#split()默认使用空格进行分割，中间无论多少
空格都切掉。
                        fd.write(text)
        print('输出成功....')
```

使用此功能需要首先建立两个文件夹，即“input.txt”和“output.txt”，并在“input.txt”输入带有空行的内容，如下图所示。



此功能的其实就是首先读取输入文件的内用，然后利用循环检测其中的空行，并使用 split 使用空格分割，将中间的空行全部切除，并将结果输出到“output.txt”，如下图所示。



我们可以发现，在第一行和第三行的空行被删除掉，实验目的达到了。

4.3 THULAC 与 jieba 对比

通过对比我们发现，从直观角度上，thulac 的分词相对来说分得更加细一点，但是这也导致其在某些时候存在一些问题，比如“深深地”可以作为一个副词，但是，其被 thulac 分为两个词。从准确率来说，根据官方统计，无论是从新闻数据(MSRA)、混合型文本(CTB8)、网络文本(WEIBO)数据上来说，thulac 的准确率明显都要高于 jieba。从速度来说，在进行运行时，可以明显感觉到其速度会明显低于 jieba，而从一些专业机构的测评来说，我们也发现确实是这样的，jieba 的运行速度差不多是 thulac 的两倍。

最后我们得出结论，在不介意速率的话，使用 thulac 进行自然语言处理（NLP）的准确率和效果相对于 jieba 的话，会更好一点。下面我们利用了 thulac，将其与词云进行联系，从 CSDN 上查询资料，从而获得其想要的云图。

4.4 使用 THULAC 生成词云

4.4.1 原理

“词云”的概念是由美国西北大学新闻学副教授里奇•戈登提出的，其核心也就是通过形成“关键词云层”或“关键词渲染”，对网络文本中出现频率较高的“关键词”的视觉上的突出。使用 python 生成词云也是目前常用的一种方法。其原理如下：

（1）.对文本数据进行分词，也是众多 NLP 文本处理的第一步，对于 wordcloud 中的 process_text（）方法，主要是停词的处理，而 THULAC 也主要是完成此过程操作。

（2）.计算每个词在文本中出现的频率，生成一个哈希表。词频计算相当于各种分布式计算平台的第一案例 wordcount， 和各种语言的 hello world 程序具有相同的地位了，呵呵。

（3）.根据词频的数值按比例生成一个图片的布局，类 IntegralOccupancyMap 是该词云的算法所在，是词云的数据可视化方式的核心。

（4）.将词按对应的词频在词云布局图上生成图片。

（5）.完成词云上各词的着色,默认是随机着色

4. 4. 2 实现

（1）对于其功能的实现首先也得像 THULAC 一样，安装词云，即

```
pip install wordcloud
```

(2)实际运行时，首先需要要建立一个 txt 文件，在文件中输入想要生成词云的文章，在这里我选用了人民日报近期刊发的一篇有关新冠疫情的新闻-《“中国在国际抗疫合作中发挥重要作用”—访世界卫生组织新冠疫情防控特使纳巴罗》，然后通过相关函数检测是否为中文，并处理成合法的标准格式。介于篇幅原因，代码在附录中有所体现。

（3）然后读取文件，调用 thulac 的分词功能，对其进行分词处理，并且获取其长度。

（4）然后通过 Wordcloud，绘制词云。其中 width,height,margin 可以设置图片属性.

```
wc = WordCloud(collocations=False, height=1400, width=1400,margin=2 ,font_path=font)
.generate(f)
```

则生成的词云如下图所示，我们可以清晰地看到，从“中国”、“疫情”、“世卫”、“合作”、“抗疫”等词中看出这篇文章的主题就是和世卫组织对中国抗疫评价相关，比较好的展现出了其核心内容。

图 1 词云图

5. 实验总结

这是我第一次正式接触 python，更是第一次接触机器学习，在使用相关代码的时候，总存在一些大大小小的问题，但是完成下来，收获还是挺大的。在使用 jieba 时，由于篇幅原因，其还有很多功能都没有呈现，比如其自定义词典功能、基于 TextRank 算法的关键词提取、并行分词等功能，但是，在稍后，我会逐渐去学习这些功能。

这次实验，我通过安装运行相关代码，对 jieba 和 THULAC 的分词功能进行了比较，其各有优势，但是 THULAC 得准确性更高，因此使用其进行了词云的生成，让我第一次体验到了 NLP 的魅力。当然，就准确性而言，目前北大已经开发出了 PKUSEG，根据官方数据，其分词准确率远高于 jieba 和 THULAC，这也正是我们以后需要探索学习实用的功能包。

附录：

词云代码：

```
import thulac
import sys
import io
import math
import numpy
from os import path
from wordcloud import WordCloud, ImageColorGenerator
import matplotlib.pyplot as plt
from PIL import Image
from matplotlib.pyplot import imread
sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding='utf8') #改变标准输出的默认编码
'''
    检查是否为中文字符
'''
def check_contain_chinese(check_str):
    for ch in check_str:
        if u'\u4e00' <= ch <= u'\u9fff':
            return True
        return False

'''
    数据清洗并改成标准格式，使用生成器
'''
def createGenerator(mylist):
    for word in mylist:
        papapa = True
        for check in word[0]:
            biubiu = check_contain_chinese(check)
            if(biubiu):
                pass
            else:
                papapa = False
                break
        if(papapa):
            yield word[0]
```

```
'''
    读取文件，进行分词处理并获取长度
'''
def CutArticle(article):
    file = open(article,'rb')
    data = file.read().decode('utf-8')
```

```
    file.close()
    thu = thulac.thulac()
    text = thu.cut(data)
    length = len(text)
    demo = createGenerator(text)
    return demo,length
if __name__ == '__main__':
    mygenerator,length = CutArticle("./1.txt")
    #print(mygenerator)
    listkey = []
    for key in mygenerator:
        listkey.append(key)
    f = ' '.join(listkey)
    print(f)
    #back_coloring = imread("./bg.jpg")
    font =r"./1.ttf"
    wc = WordCloud(collocations=False, height=1400, width=1400,margin=2 ,font_path=font).gener
ate(f)
    # width,height,margin 可以设置图片属性
    # generate 可以对全部文本进行自动分词,但是他对中文支持不好
    # 你可以通过font_path 参数来设置字体集
    #background_color 参数为设置背景颜色,默认颜色为黑色
    #image_colors = ImageColorGenerator(back_coloring)
    # 显示图片
    plt.imshow(wc)
    # 关闭坐标轴
    plt.axis('off')
    # 绘制词云
    plt.figure()
    #plt.imshow(wc.recolor(color_func=image_colors))
    plt.axis('off')
    # 保存图片
    wc.to_file('./1.png')
    # 保存图片,但是在第三模块的例子中 图片大小将会按照 mask 保存
```