中文文本挖掘及部分内容的python实现

主要方向

- 文本结构分析
- 文本摘要
- 文本聚类
- 文本分类
- 文本关联性分析
- 分布分析和趋势预测

主要流程

1. 数据预处理
    i. 文本分词
    ii. 去停用词
2. 提取关键词
3. 主题模型分析 文本聚类 文本分类
4. 文本相似度
5. 情感分析 观点挖掘

jieba

1. 分词
2. 词性标注
3. 基于TF-IDF算法的关键词抽取
4. 基于 TextRank 算法的关键词抽取
5. Tokenize：返回词语在原文的起止位置

分词

1. 全模式

2. 精确模式

3. 搜索引擎模式

```
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))
```

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

【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，
```

HMM模型，新词发现，使用Viterbi算法

可以设置用户自定义词典

词性标注

```
>>>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
```

关键词抽取

```
#TF-IDF
import jieba.analyse
jieba.analyse.extract_tags(str(content), topK=20, withWeight=False, allowPOS=())
#output：
产品 用户 经理 功能 问题 团队 数据 场景 体验 互联网 项目 内容 文档 时间 公司 信息 业务 过程 流程 页面 技术
##stopwords
 jieba.analyse.set_stop_words(file_name)

#TextRank
jieba.analyse.textrank(sentence, topK=100, withWeight=False, allowPOS=('ns', 'n', 'vn', 'v'
```

TextRank: Bringing Order into Texts

词云 wordcloud

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image

#文本数据
text=open('/Users/pone/Documents/文本分析/分词/分词结果.txt').read()

#设置背景
masks=np.array(Image.open('/Users/pone/Pictures/wine-2891894__340.jpg'))
wc=WordCloud(background_color='white', #背景色
             mask=masks, #背景
             max_words=300, #字数最多300
             font_path='汉仪火柴体简.ttf',#设置字体，默认为英文
             max_font_size=200, #最大字号
             random_state=30)#随机状态，即配色方案

#生成
myword=wc.generate(text)

#图的显示
plt.figure(1)
plt.imshow(myword)
plt.axis('off')
plt.show()
```

Github参考

用基于TDIDF提取的Top20关键词画出的词云图

LDA主题模型分析

LDA是一种典型的无监督（也就是每段文本没有标签，我们事先不知道里面说的是啥）、基于统计学习的词袋模型，即它认为一篇文档是由一组词构成的一个集合，词与词之间没有顺序以及先后的关系。一篇文档可以包含多个主题，文档中每一个词都由其中的一个主题生成。文档是若干主题的混合概率分布，而每个主题又是一个关于单词的混合概率分布。

常用的实现lda的包

- lda
- gensim
- sklearn

```
## 生成Document-word matrix

from sklearn.feature_extraction.text import CountVectorizer

tf_vectorizer = CountVectorizer()

docs=open('/Users/pone/Documents/文本分析/分词/分词结果.txt','r').read().split('\n')

X = tf_vectorizer.fit_transform(docs)
```

```
#n_topics设置主题数，n_iter设置迭代次数
model=lda.LDA(n_topics=5,n_iter=500,radom_state=1)
model.fit(X)


#topic-word分布
topic_word=model.topic_word_
print('type(topic_word):{}'.format(type(topic_word)))
print("shape:{}".format(topic_word.shape))
print(topic_word[1:2,:3])
#topic_word[] 第一个参数为主题索引，第二个参数为词索引

#Top-N单词
n=5
for i,topic in enumerate(topic_word):
    topic_words=np.array(vocab)[np.argsort(topic)][:-(n+1):-1]
    print('*Topic {}\n- {}'.format(i, ' '.join(topic_words)))
#doc-topic
oc_topic=model.doc_topic_
print("type(doc_topic): {}".format(type(doc_topic)))
print("shape: {}".format(doc_topic.shape))

for n in range(10):
    topic_most_pr = doc_topic[n].argmax()
    print("doc: {} topic: {}".format(n, topic_most_pr))
```

```
*Topic 0
- 用户 产品 需求 功能 使用
*Topic 1
- 产品经理 产品 没有 问题 很多
*Topic 2
- 平台 互联网 行业 公司 服务
*Topic 3
- 功能 用户 数据 需要 进行
*Topic 4
- 产品 需求 项目 需要 团队
type(doc_topic): <class 'numpy.ndarray'>
shape: (1200, 5)
doc: 如何用一句话证明你是做产品的？ topic: 1
doc: 关于新零售，我有这么一个想法 topic: 2
doc: 产品经理，要“看懂”、更要“看破” topic: 0
doc: UX冲刺：Google大神手把手教你Storyboard topic: 3
doc: 互联网产品思维：怎么解决看病就医更靠谱？ topic: 2
doc: 经验分享 | 详解产品实现的五大过程 topic: 4
doc: 合格产品经理必须“懂”系列（1）：懂项目管理 topic: 4
doc: 作为产品经理，你有进行自检吗？ topic: 1
doc: 一个产品的成功与否，和范围管理有直接的关系 topic: 4
doc: 产品从0到1，该考虑哪些维度？ topic: 0
```

情感极性分析

SnowNLP

- 中文分词（Character-Based Generative Model）
- 词性标注（TnT 3-gram 隐马）
- 情感分析（现在训练数据主要是买卖东西时的评价，所以对其他的一些可能效果不是很好，待解决）
- 文本分类（Naive Bayes）
- 转换成拼音（Trie树实现的最大匹配）
- 繁体转简体（Trie树实现的最大匹配）
- 提取文本关键词（TextRank算法）
- 提取文本摘要（TextRank算法）
- tf，idf
- Tokenization（分割成句子）
- 文本相似（BM25）

```
s=SnowNLP(u'康老师真漂亮')
print(s.words)
a=s.tags
for i in a:
    print(i)
print(s.sentiments)
```

输出

```
['康', '老师', '真', '漂亮']
('康', 'nr')
('老师', 'n')
('真', 'd')
('漂亮', 'a')
0.9765966076488548
```

词频统计

Packages:

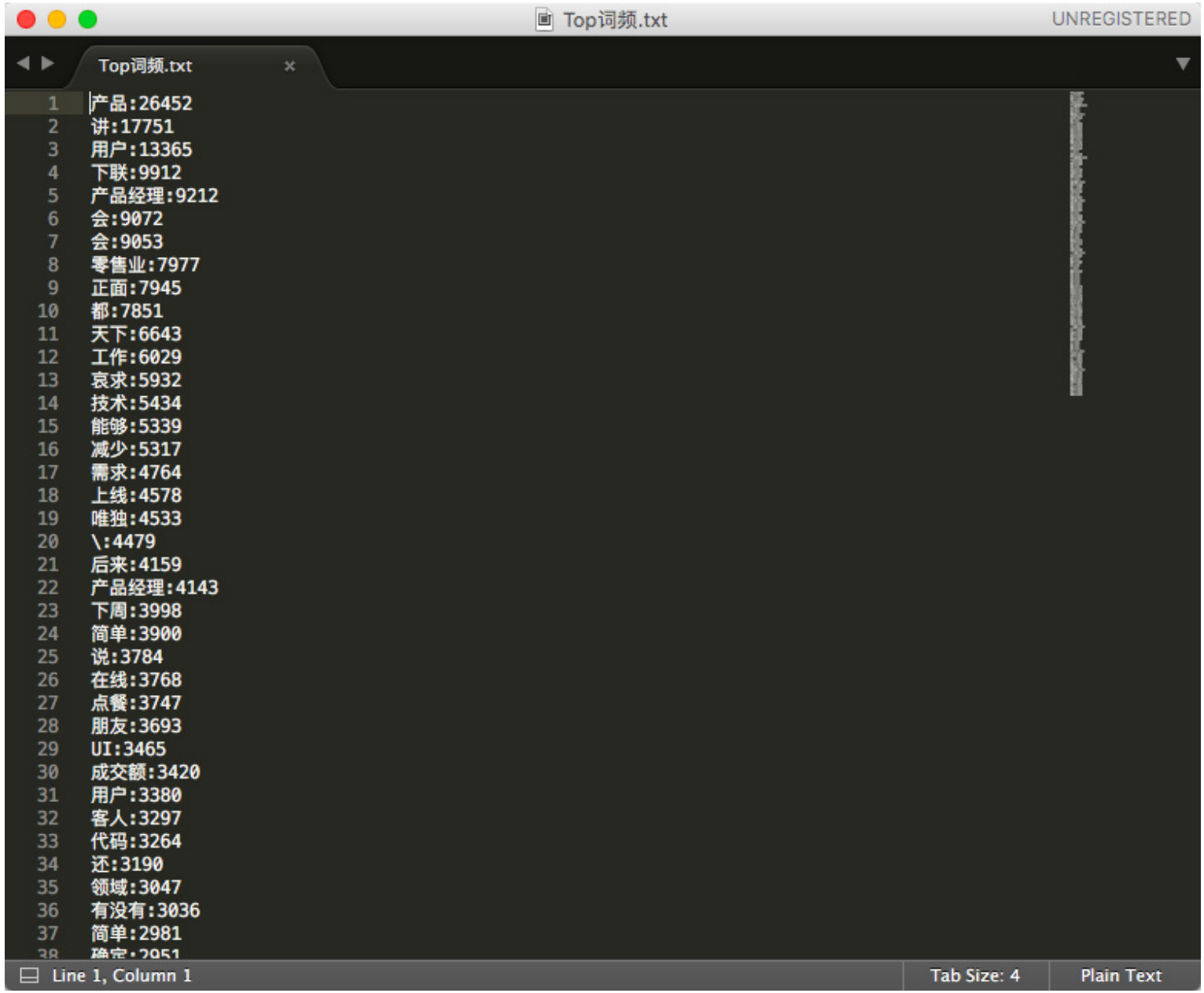
1. collections Counter
2. argsort

```
def Count_Save(All_docs,top):
    f = open('/Users/pone/Documents/文本分析/分词/Top词频.txt','w')
    Number_dict=Counter(All_docs)

    Number_list=list(Number_dict.values())

#从小到大排序好的索引列表
    Number_list_array=np.array(Number_list).argsort()

    for i in range(1,top):
        f.write(All_docs[Number_list_array[-i]])
        f.write(':')
        f.write(str(Number_list[Number_list_array[-i]]))
        f.write('\n')
    f.close()
Count_Save(All_docs,100)
```


Top词频.txt
UNREGISTERED
Top词频.txt
1 产品:26452
2 讲:17751
3 用户:13365
4 下联:9912
5 产品经理:9212
6 会:9072
7 会:9053
8 零售业:7977
9 正面:7945
10 都:7851
11 天下:6643
12 工作:6029
13 哀求:5932
14 技术:5434
15 能够:5339
16 减少:5317
17 需求:4764
18 上线:4578
19 唯独:4533
20 \:4479
21 后来:4159
22 产品经理:4143
23 下周:3998
24 简单:3900
25 说:3784
26 在线:3768
27 点餐:3747
28 朋友:3693
29 UI:3465
30 成交额:3420
31 用户:3380
32 客人:3297
33 代码:3264
34 还:3190
35 领域:3047
36 有没有:3036
37 简单:2981
Line 1, Column 1
Tab Size: 4
Plain Text