这是《Python数据挖掘课程》系列文章，前面很多文章都讲解了数据挖掘、机器学习，这篇文章主要讲解LDA和pyLDAvis算法，同时讲解如何读取CSV文本内容进行主题挖掘及可视化展示。
文章比较基础，希望对你有所帮助，提供些思路，也是自己教学的内容。推荐大家购买作者新书《Python网络数据爬取及分析从入门到精通（分析篇）》，如果文章中存在错误或不足之处，还请海涵。

该系列文章代码&数据集下载地址：https://github.com/eastmountyxz/Python-for-Data-Mining
希望读者能帮Github点个赞，一起加油。

目录：

一.显示结果及安装
二.LDA主题挖掘
三.pyLDAvis可视化分析
四.小结

前文参考：

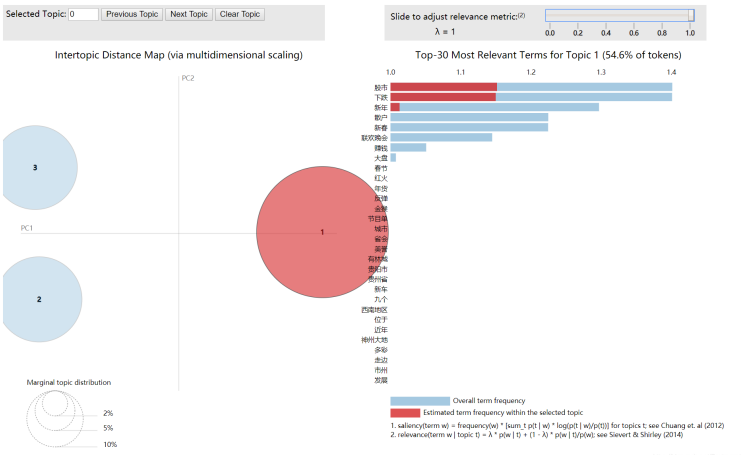
【Python数据挖掘课程】一.安装Python及爬虫入门介绍
【Python数据挖掘课程】二.Kmeans聚类数据分析及Anaconda介绍
【Python数据挖掘课程】三.Kmeans聚类代码实现、作业及优化
【Python数据挖掘课程】四.决策树DTC数据分析及鸢尾数据集分析
【Python数据挖掘课程】五.线性回归知识及预测糖尿病实例
【Python数据挖掘课程】六.Numpy、Pandas和Matplotlib包基础知识
【Python数据挖掘课程】七.PCA降维操作及subplot子图绘制
【Python数据挖掘课程】八.关联规则挖掘及Apriori实现购物推荐
【Python数据挖掘课程】九.回归模型LinearRegression简单分析氧化物数据
【python数据挖掘课程】十.Pandas、Matplotlib、PCA绘图实用代码补充
【python数据挖掘课程】十一.Pandas、Matplotlib结合SQL语句可视化分析
【python数据挖掘课程】十二.Pandas、Matplotlib结合SQL语句对比图分析
【python数据挖掘课程】十三.WordCloud词云配置过程及词频分析
【python数据挖掘课程】十四.Scipy调用curve_fit实现曲线拟合
【python数据挖掘课程】十五.Matplotlib调用imshow()函数绘制热图
【python数据挖掘课程】十六.逻辑回归LogisticRegression分析鸢尾花数据
【python数据挖掘课程】十七.社交网络Networkx库分析人物关系（初识篇）
【python数据挖掘课程】十八.线性回归及多项式回归分析四个案例分享
【python数据挖掘课程】十九.鸢尾花数据集可视化、线性回归、决策树花样分析
【python数据挖掘课程】二十.KNN最近邻分类算法分析详解及平衡秤TXT数据集读取
【python数据挖掘课程】二十一.朴素贝叶斯分类器详解及中文文本舆情分析
【python数据挖掘课程】二十二.Basemap地图包安装入门及基础知识讲解
【python数据挖掘课程】二十三.时间序列金融数据预测及Pandas库详解

【python数据挖掘课程】二十四.KMeans文本聚类分析互动百科语料
【python数据挖掘课程】二十五.Matplotlib绘制带主题及聚类类标的散点图
【python数据挖掘课程】二十六.基于SnowNLP的豆瓣评论情感分析
【python数据挖掘课程】二十七.基于SVM分类器的红酒数据分析

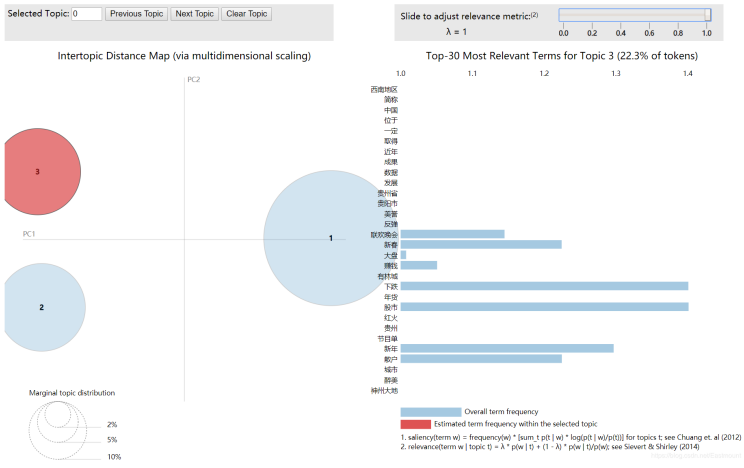
一.显示结果及安装

参考LDA前文：
[python] LDA处理文档主题分布及分词、词频、tfidf计算
[python] LDA处理文档主题分布代码入门笔记

调用LDA和pyLDAvis的运行结果如下图所示，将文本数据划分为3个主题，每个主题对应相关的关键词及比例。当选择第一个主题时，它显示为红色，其中“股市”、“下跌”表示对应的关键词出现概率。

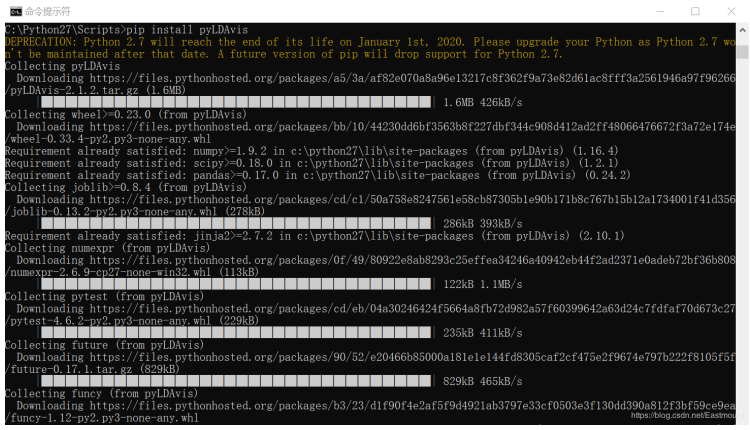


其他主题也类似，如下图所示：



本文和之前介绍的LDA算法略有不同，它主要采用sklearn中的LatentDirichletAllocation包实现主题分布研究，并调用pyLDAvis绘制相关图形。安装过程如下所示：

```
pip install pyLDAvis
```



二.LDA主题挖掘

1.数据读取

本文数据集共15行，涉及新春、股市和贵州三个主题，如下图所示：

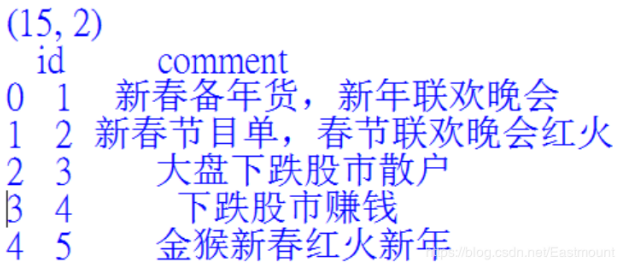
	A	B
1	id	comment
2	1	新春备年货，新年联欢晚会
3	2	新春节目单，春节联欢晚会红火
4	3	大盘下跌股市散户
5	4	下跌股市赚钱
6	5	金猴新春红火新年
7	6	新车新年年货新春
8	7	股市反弹下跌
9	8	股市散户赚钱
10	9	新年,看春节联欢晚会
11	10	大盘下跌散户
12	11	贵州省位于中国西南地区，简称黔
13	12	走边神州大地，醉美多彩贵州
14	13	贵阳市是贵州省省会城市，有林城的美誉
15	14	贵州省包括九个市州和一个新区
16	15	贵阳市近年发展大数据取得一定成果

```
id  comment
1   新春备年货，新年联欢晚会
2   新春节目单，春节联欢晚会红火
3   大盘下跌股市散户
4   下跌股市赚钱
5   金猴新春红火新年
6   新车新年年货新春
7   股市反弹下跌
8   股市散户赚钱
9   新年,看春节联欢晚会
10  大盘下跌散户
11  贵州省位于中国西南地区，简称黔
12  走边神州大地，醉美多彩贵州
13  贵阳市是贵州省省会城市，有林城的美誉
14  贵州省包括九个市州和一个新区
15  贵阳市近年发展大数据取得一定成果
```

调用pandas读取CSV文件核心代码如下：

```
#coding: utf-8
import pandas as pd

#读取数据
f = open('data3.csv')
df = pd.read_csv(f)
print(df.shape)          #查看数据维度
print(df.head())         #查看前几行数据
```

输出结果如下所示：



2.Jieba中文分词

在进行文本挖掘之前需要进行中文分词处理，下面是调用Jieba工具进行分词和词性过滤的代码。

```
import jieba
import jieba.posseg as psg

a = '我想大口吃肉喝酒看电影，质量真差,好漂亮啊'
sen = psg.cut(a)
for n in sen:
    print n

#过滤词性
def chinese_word_cut(mytext):
    result = psg.cut(mytext)
    return ' '.join(x.word for x in result if x.flag =='a' or x.flag =='r

print(chinese_word_cut(a))
```

输出结果如下所示：

```
Building prefix dict from the default dictionary ...
Loading model from cache c:\users\yxz\appdata\local\temp\jieba.cache
Loading model cost 0.466 seconds.
Prefix dict has been built succesfully.
我/r
想/v
大/a
口吃/n
肉/n
喝酒/v
看/v
电影/n
，/x
质量/n
真差/n
,/x
好/a
漂亮/a
啊/zg
想 大 口吃 肉 喝酒 看 电影 质量 真差 好 漂亮
```


接着对CSV文本进行中文分词处理，核心代码如下：

```
#coding: utf-8
import pandas as pd

#第一步 读取数据
f = open('data3.csv')
df = pd.read_csv(f)
print(df.shape)          #查看数据维度
print(df.head())         #查看前几行数据

#第二步 中文分词
import jieba
import jieba.posseg as psg

#格式转换 否则会报错  'float' object has no attribute 'decode'
df = pd.DataFrame(df['comment'].astype(str))

def chinese_word_cut(mytext):
    return ' '.join(jieba.cut(mytext))

#增加一列数据
df['content_cutted'] = df['comment'].apply(chinese_word_cut)
print df.content_cutted.head()
```

此时的输出结果如下所示：

```
0       新春 备 年货 ， 新年 联欢晚会
1    新春 节目单 ， 春节 联欢晚会 红火
2            大盘 下跌 股市 散户
3               下跌 股市 赚钱
```

```
4              金猴 新春 红火 新年
Name: content_cutted, dtype: object
```

3.词频及TF-IDF计算

代码如下:

```
#coding: utf-8
import pandas as pd

#第一步 读取数据
f = open('data3.csv')
df = pd.read_csv(f)
print(df.shape)          #查看数据维度
print(df.head())         #查看前几行数据

#第二步 中文分词
import jieba
import jieba.posseg as psg

#格式转换 否则会报错  'float' object has no attribute 'decode'
df = pd.DataFrame(df['comment'].astype(str))

def chinese_word_cut(mytext):
    return ' '.join(jieba.cut(mytext))

#增加一列数据
df['content_cutted'] = df['comment'].apply(chinese_word_cut)
print df.content_cutted.head()

#第三步 计算TF-IDF值
from sklearn.feature_extraction.text import TfidfVectorizer, CountVector

#设置特征数
n_features = 2000

tf_vectorizer = TfidfVectorizer(strip_accents = 'unicode',
                                max_features=n_features,
                                stop_words=['的','或','等','是','有','之',
                                           '一个','和','也','被','吗','于'
                                           '一下','几天','200','还有','一看
                                            '“','”','。',',',' ','？','、','
                                            'and','in','of','the','我们'
                                          '了','有些','已经','不是','这么',
                                          '一种','位于','之一','天空','没有
                                          '特别'],
```

```
                                    max_df = 0.99,
                                    min_df = 0.002) #去除文档内出现几率过大或过小
tf = tf_vectorizer.fit_transform(df.content_cutted)

print(tf.shape)
print(tf)
```

此时输出结果如下，包括15行数据，41个特征词，并计算每个特征词的TF-IDF值。
推荐作者前文：[python] 使用scikit-learn工具计算文本TF-IDF值

```
(15, 41)
  (0, 29)  0.5064286281539676
  (0, 19)  0.4590722397658956
  (0, 14)  0.5674816313668122
  (0, 20)  0.4590722397658956
  (1, 27)  0.4575288281883257
  (1, 22)  0.4575288281883257
  (1, 31)  0.5269056459698538
  (1, 29)  0.4083051926143108
  (1, 20)  0.3701243746127834
  (2, 16)  0.5064286281539676
```

4.LDA主题挖掘

进行LDA主题挖掘，计算各个主题及对应关键词的核心代码如下：

```
#coding: utf-8
import pandas as pd

#------------------  第一步 读取数据  ------------------------
f = open('data3.csv')
df = pd.read_csv(f)
print(df.shape)          #查看数据维度
print(df.head())         #查看前几行数据

#------------------  第二步 中文分词  ----------------------
import jieba
import jieba.posseg as psg

#格式转换 否则会报错  'float' object has no attribute 'decode'
df = pd.DataFrame(df['comment'].astype(str))
```

```
def chinese_word_cut(mytext):
    return ' '.join(jieba.cut(mytext))

#增加一列数据
df['content_cutted'] = df['comment'].apply(chinese_word_cut)
print df.content_cutted.head()

#------------------  第三步 计算TF-IDF值  ---------------------
from sklearn.feature_extraction.text import TfidfVectorizer, CountVector

#设置特征数
n_features = 2000

tf_vectorizer = TfidfVectorizer(strip_accents = 'unicode',
                                max_features=n_features,
                                stop_words=['的','或','等','是','有','之',
                                           '一个','和','也','被','吗','于'
                                           '一下','几天','200','还有','一看
                                            '“','”','。',',','？','、','
                                            'and','in','of','the','我们'
                                          '了','有些','已经','不是','这么',
                                          '一种','位于','之一','天空','没有
                                          '特别'],
                                max_df = 0.99,
                                min_df = 0.002) #去除文档内出现几率过大或过小
tf = tf_vectorizer.fit_transform(df.content_cutted)

print(tf.shape)
print(tf)

#---------------------  第四步 LDA分析  ----------------------
from sklearn.decomposition import LatentDirichletAllocation

#设置主题数
n_topics = 3

lda = LatentDirichletAllocation(n_topics=n_topics,
                                max_iter=100,
                                learning_method='online',
                                learning_offset=50,
                                random_state=0)
lda.fit(tf)

#显示主题数 model.topic_word_
print(lda.components_)
#几个主题就是几行 多少个关键词就是几列
print(lda.components_.shape)
```

```
#计算困惑度
print(u'困惑度：')
print lda.perplexity(tf,sub_sampling = False)

#主题-关键词分布
def print_top_words(model, tf_feature_names, n_top_words):
    for topic_idx,topic in enumerate(model.components_):    # lda.compone
        print('Topic #%d:' % topic_idx)
        print(' '.join([tf_feature_names[i] for i in topic.argsort()[:-n_
        print("")

#定义好函数之后 暂定每个主题输出前20个关键词
n_top_words = 20
tf_feature_names = tf_vectorizer.get_feature_names()
#调用函数
print_top_words(lda, tf_feature_names, n_top_words)
```

输出的三个主题对应关键词的概率如下，形状为（3L, 41L）。

```
[[0.75895885 0.34444982 0.34973918 0.34295623 0.75608541 0.34497434
  0.75747747 0.34648015 0.34465879 0.34438872 0.3479937  0.78019007
  0.34228347 0.75711457 0.6070849  0.34188516 0.34880518 0.3462341
  0.75500429 0.4423738  0.46351765 0.87794869 0.34453203 0.3435454
  0.3448315  0.77963258 0.34203442 0.34561508 0.34424401 0.3472218
  0.3462053  0.34637752 0.34335099 0.77977343 0.67175953 0.34439304
  0.34395113 0.78103352 0.34498333 0.78056264 0.34649137]
 [0.34253366 0.34290794 2.31277487 0.3420248  0.34490947 0.34433589
  0.34142761 1.03539345 0.34318926 0.34280508 0.76315118 0.34417968
  1.52608392 0.34371958 1.15340909 0.34163671 1.95986875 0.34283976
  0.34253935 2.03731915 1.87391339 0.39995597 1.41775637 0.76242987
  0.76305792 0.34422024 0.34206921 1.30442141 0.76302297 1.79761514
  2.31640706 0.85446292 0.34437878 0.34275433 0.64960775 0.69595576
  1.61208815 0.34386861 0.34430167 0.34235254 0.93306227]
 [0.3441964  0.72214725 0.34648086 0.79848879 0.34275468 0.79851923
  0.34362251 0.34986917 0.71778923 0.72079642 0.3450307  0.34227886
  0.34775162 0.34483406 0.34639127 0.72074042 0.34509185 0.71843938
  0.34392252 0.34537485 0.34857924 0.34446653 0.3445893  0.34669583
  0.34462607 0.3448961  0.80027236 0.34575025 0.35023403 0.34876591
  0.34587198 0.34544261 0.80093901 0.34549286 0.71537749 0.68397861
  0.34720577 0.34428762 0.72080161 0.34493498 0.34444472]]
```


困惑度及各个主题下的关键词通过for循环显示，如下：

```
困惑度：
146.58072228209318

Topic #0:
新车 走边 醉美 多彩 贵州 神州大地 一个 包括 市州 九个 新区 贵州省
年货 新春 新年 下跌 散户 城市 联欢晚会 金猴

Topic #1:
股市 下跌 新年 散户 新春 联欢晚会 赚钱 大盘 春节 红火 年货 反弹 金猴
节目单 城市 省会 美誉 有林城 贵阳市 贵州省
```

```
Topic #2:
西南地区 简称 位于 中国 一定 近年 取得 成果 数据 发展 贵州省
贵阳市 美誉 反弹 联欢晚会 新春 大盘 赚钱 有林城 下跌
```

三.pyLDAvis可视化分析

最后补充调用pyLDAvis进行可视化分析的完整代码，如下所示：

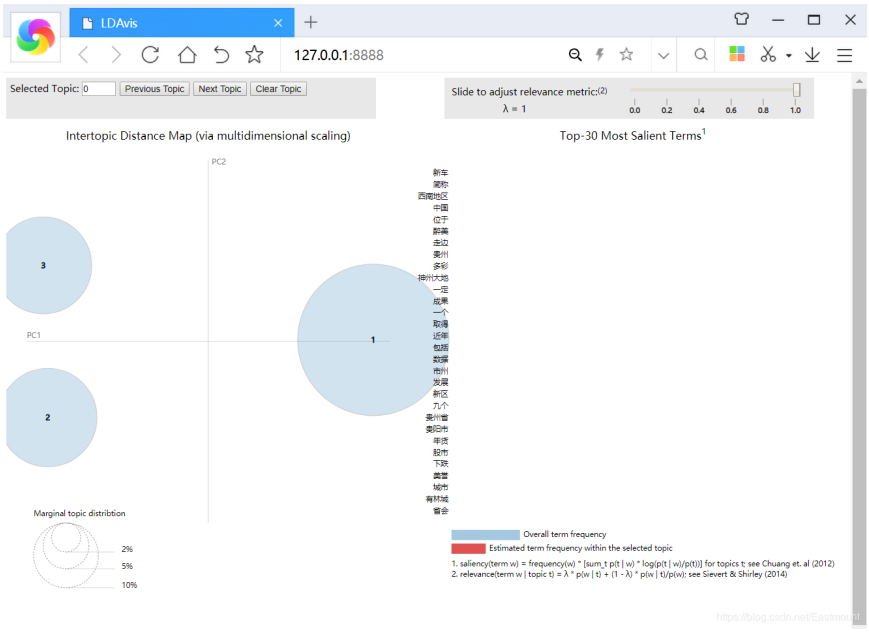
最终代码

```
#coding: utf-8
import pandas as pd

#------------------  第一步 读取数据  ------------------------
f = open('data3.csv')
df = pd.read_csv(f)
print(df.shape)          #查看数据维度
print(df.head())         #查看前几行数据

#------------------  第二步 中文分词  ------------------------
import jieba
import jieba.posseg as psg

#格式转换 否则会报错  'float' object has no attribute 'decode'
df = pd.DataFrame(df['comment'].astype(str))

def chinese_word_cut(mytext):
    return ' '.join(jieba.cut(mytext))

#增加一列数据
df['content_cutted'] = df['comment'].apply(chinese_word_cut)
print df.content_cutted.head()

#------------------  第三步 计算TF-IDF值  --------------------
from sklearn.feature_extraction.text import TfidfVectorizer, CountVector

#设置特征数
n_features = 2000

tf_vectorizer = TfidfVectorizer(strip_accents = 'unicode',
                                max_features=n_features,
                                stop_words=['的','或','等','是','有','之',
                                           '一个','和','也','被','吗','于'
                                           '一下','几天','200','还有','一
```

```
                                                '“','”','。',',','，','？','、','
                                                'and','in','of','the','我们'
                                               '了','有些','已经','不是','这么',
                                               '一种','位于','之一','天空','没有
                                               '特别'],
                                    max_df = 0.99,
                                    min_df = 0.002) #去除文档内出现几率过大或过小
tf = tf_vectorizer.fit_transform(df.content_cutted)

print(tf.shape)
print(tf)

#----------------------  第四步 LDA分析  ----------------------
from sklearn.decomposition import LatentDirichletAllocation

#设置主题数
n_topics = 3

lda = LatentDirichletAllocation(n_topics=n_topics,
                                    max_iter=100,
                                    learning_method='online',
                                    learning_offset=50,
                                    random_state=0)
lda.fit(tf)

#显示主题数 model.topic_word_
print(lda.components_)
#几个主题就是几行 多少个关键词就是几列
print(lda.components_.shape)

#计算困惑度
print(u'困惑度: ')
print lda.perplexity(tf,sub_sampling = False)

#主题-关键词分布
def print_top_words(model, tf_feature_names, n_top_words):
    for topic_idx,topic in enumerate(model.components_):    # lda.compone
        print('Topic #%d:' % topic_idx)
        print(' '.join([tf_feature_names[i] for i in topic.argsort()[:-n_
        print("")

#定义好函数之后 暂定每个主题输出前20个关键词
n_top_words = 20
tf_feature_names = tf_vectorizer.get_feature_names()
#调用函数
print_top_words(lda, tf_feature_names, n_top_words)
```

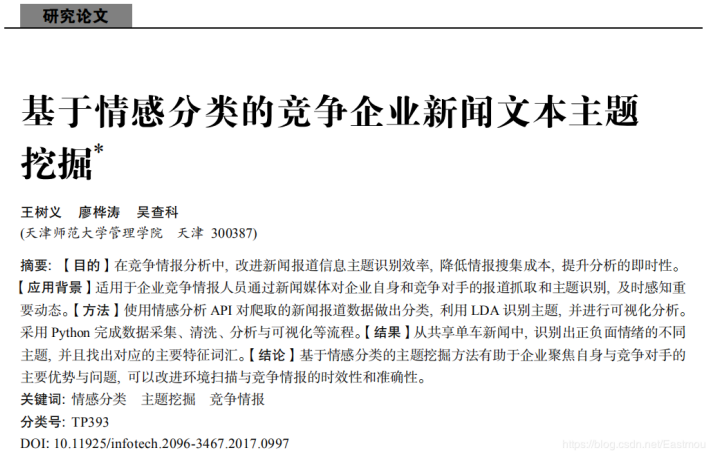
```
#------------------   第五步 可视化分析  ---------------------
import pyLDAvis
import pyLDAvis.sklearn

#pyLDAvis.enable_notebook()

data = pyLDAvis.sklearn.prepare(lda,tf,tf_vectorizer)
print(data)

#显示图形
pyLDAvis.show(data)

#pyLDAvis.save_json(data,' fileobj.html')
```

浏览器打开如下所示：



四.小结

该篇文章主要讲解了基于LDA和pyLDAvis的主题挖掘及可视化分析，属于文本挖掘、数据挖掘、主题分析的基础性文章，并且推荐同学们结合作者之前LDA代码进行理解。同时，推荐大家看看王树义老师的论文，尝试相关的文章。



研究论文

基于情感分类的竞争企业新闻文本主题挖掘*

王树义 廖梓涛 吴查科
(天津师范大学管理学院 天津 300387)

摘要: 【目的】在竞争情报分析中，改进新闻报道信息主题识别效率，降低情报搜集成本，提升分析的即时性。【应用背景】适用于企业竞争情报人员通过新闻媒体对企业自身和竞争对手的报道抓取和主题识别，及时感知重要动态。【方法】使用情感分析 API 对爬取的新闻报道数据做出分类，利用 LDA 识别主题，并进行可视化分析。采用 Python 完成数据采集、清洗、分析与可视化等流程。【结果】从共享单车新闻中，识别出正负面情绪的不同主题，并且找出对应的主要特征词汇。【结论】基于情感分类的主题挖掘方法有助于企业聚焦自身与竞争对手的主要优势与问题，可以改进环境扫描与竞争情报的时效性和准确性。

关键词: 情感分类 主题挖掘 竞争情报
分类号: TP393
DOI: 10.11925/infotech.2096-3467.2017.0997

同时该篇文章需要注意几点：

1.感觉采用lda包进行主题挖掘的准确性更高，而Sklearn中的LatentDirichletAllocation算法效果不理想，它会将各主题关键词混淆。但是画图好像又需要和Sklearn结合。

2.推荐博友们看看主题挖掘、主题演化相关的论文，尤其是南大核心，对研究生的毕业论文有一定帮助。

3.作者需要学习的东西太多，总感觉自己有些夸夸其谈，希望未来博士期间，能静下心来，真正分享一些比较好的文章，做点科研。同时，深入做些成果，而不是还是这些表象。

最后希望这篇文章对你有所帮助，且分享且珍惜，共勉。
(By:Eastmount 2019-06-12 中午13点 http://blog.csdn.net/eastmount/)