

第八届“泰迪杯”

全国大学生数据挖掘挑战赛

论

文

作品名称：“智慧政务”中的文本挖掘应用（C 题）

“智慧政务”中的文本挖掘应用

摘要

近年来，随着微信、微博、市长信箱、阳光热线等网络问政平台逐步成为政府了解民意、汇聚民智、凝聚民气的重要渠道，各类社情民意相关的文本数据量不断攀升，给以往主要依靠人工来进行留言划分和热点整理的相关部门的工作带来了极大挑战。同时，随着大数据、云计算、人工智能等技术的发展，建立基于自然语言处理技术的智慧政务系统已经是社会治理创新发展的新趋势，对提升政府的管理水平和施政效率具有极大的推动作用。

针对问题一：群众留言的分类问题，先进行分类标签的定义以及数据的简单清洗，然后进行数据的读取与抽取。采用当前广泛使用的基于 Python 的中文分词工具 jieba 对文本进行分词，去停用词来降低句子噪音对句子的理解，减少特征词的数量。根据词云图结果来修改停用词的去用来提炼数据，提高数据的有效性。分类方法是利用 sklearn 实现多分类 demo，引入相应的库，进行加载数据及数据格式转化，建立训练模型，最后进行相应的性能评估。

针对问题二：热点问题挖掘，先对数据进行简单的清洗，可以先将留言编号和留言用户以及留言时间这三列无用数据删除，然后进行数据的读取与抽取。绘制词云图，再用 jieba 对文本进行分词，从而找到热点问题。找到热点问题后，热度指数可以用该问题出现的次数来评价，然后将热点问题进行排列。

针对问题三：答复意见的评价，从答复的相关性、完整性、可解释性等角度指出答复意见的不足并给出相关的建议。

关键词：文本分类；数据清洗；词云图；分词；数据特征

Absrtact

in recent years, with WeChat, Weibo, Mayor's mailbox, Sunshine Hotline and other online political platforms gradually becoming an important channel for the government to understand public opinion, gather people's wisdom, and gather people's spirit, the amount of text data related to various kinds of social sentiment and public opinion has been rising, which has brought great challenges to the work of relevant departments which mainly rely on manual message division and hot spot sorting in the past. At the same time, with the development of big data, cloud computing, artificial intelligence and other technologies, the establishment of intelligent government system based on natural language processing technology has been a new trend of social governance innovation and development, which has a great role in promoting the management level and governance efficiency of the government.

For the problem one: the classification of mass messages, the first classification label definition and data cleaning, and then data reading and extraction. Using the widely used Python based Chinese word segmentation tool to jieba text segmentation, to stop words to reduce sentence noise to understand the sentence, reduce the number of feature words. According to the results of word cloud graph, we modify the removal of stop words to refine the data and improve the validity of the data. The classification method is to use sklearn to realize multi-classification demo, to introduce corresponding library, to transform loading data and data format, to establish training model, the most after the corresponding performance evaluation.

Problem 2: hot issues mining, the first simple data cleaning, you can first message number and message users and message time these three columns of useless data deleted, and then data reading and extraction. Draw the word cloud map, and then use the jieba to participle the text, so as to find hot issues. After finding hot issues, the heat index can be evaluated by the number of times the problem appears, and then the hot issues are arranged.

In response to question 3: Evaluation of response comments, from the point of

view of relevance, completeness, interpretability, etc., to point out the shortcomings of response comments and give relevant recommendations.

Keywords: text classification; data cleaning; word cloud; word segmentation; data features

目 录

1. 挖掘背景与目标	3
1.1 挖掘背景	3
1.2 挖掘目标	3
1.3 挖掘流程	4
2. 问题一：群众留言分类	4
2.1 数据预处理	4
2.2 分词.....	7
2.3 TF-IDF 的特征值.....	10
2.4 分类器的选择	13
2.5 模型的选择	14
2.6 模型的评估	15
3. 问题二：热点问题挖掘	19
3.1 数据预处理	20
3.2 分词.....	20
3.3 基于相似度的自适应最优 LDA 模型.....	21
4. 问题三：答复意见的评价	25
4.1 模型训练	25
4.2 相似度计算	29
4.3 提取特征	30
4.4 提取结果	31

5. 结论	31
参考文献.....	31

“智慧政务”中的文本挖掘应用

1. 挖掘背景与目标

1.1 挖掘背景

近年来，随着微信、微博、市长信箱、阳光热线等网络问政平台逐步成为政府了解民意、汇聚民智、凝聚民气的重要渠道，各类社情民意相关的文本数据量不断攀升，给以往主要依靠人工来进行留言划分和热点整理的相关部门的工作带来了极大挑战。同时，随着大数据、云计算、人工智能等技术的发展，建立基于自然语言处理技术的智慧政务系统已经是社会治理创新发展的新趋势，对提升政府的管理水平和施政效率具有极大的推动作用。

仅用人工去完成文本挖掘在现如今可行性不高，群众留言的数据量太多，涉及的方面很广泛，将其进行相应的归类，使留言更有序，更有益于后续相关部门针对特定的问题进行解决。基于现有的技术和方法实现自动文本挖掘(Text Mining, TM)就显得尤为重要。文本自动分类(TextClassification/Text Categorization, TC)，简称为文本分类，作为智能组织和管理海量文本数据的重要方法，是自动文本挖掘研究领域的重要组成部分。文本分类方法按照预定的类别体系对用户关心的所有信息进行类别预测和判定,在很大程度上解决了信息杂乱问题。文本分类技术的研究对于信息的价值挖掘和管理利用都具有极其重要的意义。文本分类技术作为解决文本信息处理的主要手段之一,得到了研究人员的广泛关注。文本情感分类(Text Sentiment Classification, TSC)是情感分析(Sentiment Analysis, SA) 的一个重要研究分支，其涉及文本挖掘、自然语言处理、机器学习等多个研究领域，是目前一个极受关注的课题。文本情感分类可以认为是一个特殊的文本分类问题,是一个对带有情感色彩的文本进行处理、分析、评估和类别判定的过程。随着社交平台 and 媒介的兴起和繁盛，文本情感分析可以用于检测大众舆论趋向，跟踪话题热点，评估用户对事物的情感倾向，甚至发现用户情感随时间的演化规律等。

1.2 挖掘目标

我们要构建一个智能的文本挖掘模型如图 1 所示。基于各类社情民意相关的文本数据，建立识别模型，准确地进行留言划分，以解决以往主要依靠人工来进

行留言划分和热点整理的相关部门的工作的现状及困难。

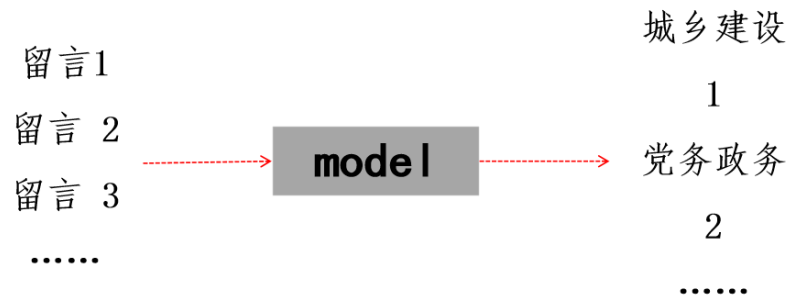


图1 文本挖掘模型

1.3 挖掘流程

在文本挖掘的过程中，先进行数据抽取和清洗将无关数据删除，再进行分词去停用词等操作可去掉很多无太大意义的词留下特征性的词，得出处理后的数据，建立文本分类的模型，再将数据进行实践，最后对所建的模型进行评价和优化。如图2所示。

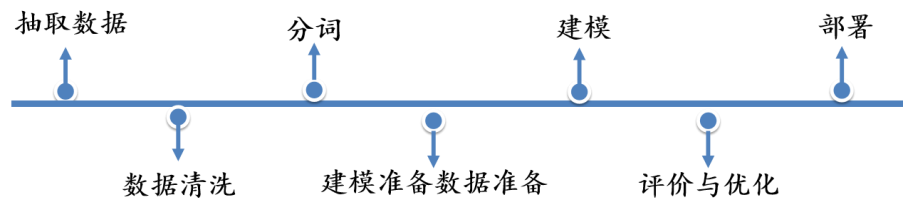


图2 挖掘流程

2. 问题一：群众留言分类

在处理网络问政平台的群众留言时，工作人员首先按照一定的划分体系（参考附件1提供的内容分类三级标签体系）对留言进行分类，以便后续将群众留言分派至相应的职能部门处理。目前，大部分电子政务系统还是依靠人工根据经验处理，存在工作量大、效率低，且差错率高等问题。请根据附件2给出的数据，建立关于留言内容的一级标签分类模型。

2.1 数据预处理

1.查看数据

查看数据，看看数据的结构，分析如何处理数据便于解题。


```
%matplotlib inline
import pandas as pd
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import jieba as jb
import re
```

```
df = pd.read_excel('f2.xlsx')
df=df[['cat','review']]
print("数据总量: %d ." % len(df))
df.sample(10)
```

得到的结果如图 3 所示。

数据总量: 9210 .

	cat	review
3668	教育文体	何时发布2019年A7县普惠性幼儿园清单及收费标准
1341	城乡建设	L4县东风西路北侧农机局一带棚户区改造项目何时动工
1025	城乡建设	K1区洒水作业时间能不能改进下
7900	商贸旅游	西地省计量局能不能提高一下效率
1560	城乡建设	M10县的电力管理乱
4649	教育文体	G市教育局教师资格证办理拖沓，严重影响孩子招考应聘
8566	卫生计生	M市M1区大汉巨龙家园有黑诊所
6036	劳动和社会保障	I1区人民织布厂老职工的社保手续办好了钱却还没落实
19	城乡建设	A5区楚府线几个小区经常停电
3351	交通运输	A3区中国邮政局网点A6区县学士所经常开不了发票

图 3 数据集

2.数据清洗

将留言主题和留言详情合并为一列，为模型建立提供更多的有效数据。将留言编号、留言用户、留言时间三列与建立模型无关数据删除，并且将一级标签换成相对应的数字。结果如图 4 所示。

```
print("在 cat 列中总共有 %d 个空值." % df['cat'].isnull().sum())
print("在 review 列中总共有 %d 个空值." %
df['review'].isnull().sum())
df[df.isnull().values==True]
df = df[pd.notnull(df['review'])]
```

在 cat 列中总共有 0 个空值。
在 review 列中总共有 0 个空值。

统计一下各个类别的数据量

```
d = {'cat':df['cat'].value_counts().index, 'count':  
df['cat'].value_counts()}  
df_cat = pd.DataFrame(data=d).reset_index(drop=True)  
df_cat
```

	cat	count
0	城乡建设	2009
1	劳动和社会保障	1969
2	教育文体	1589
3	商贸旅游	1215
4	环境保护	938
5	卫生计生	877
6	交通运输	613

图 4 数据清洗结果

3.将 cat 类转换成 id

接下来我们要将 cat 类转换成 id，这样便于以后的分类模型的训练。

我们将 cat 转换成了 Id(0 到 9),由于我们的评价内容都是中文,所以要对中文进行一些预处理工作,这包括删除文本中的标点符号,特殊符号,还要删除一些无意义的常用词(stopword),因为这些词和符号对系统分析预测文本的内容没有任何帮助,反而会增加计算的复杂度和增加系统开销,所有在使用这些文本数据之前必须要将它们清理干净。

```
df['cat_id'] = df['cat'].factorize()[0]  
cat_id_df = df[['cat',  
'cat_id']].drop_duplicates().sort_values('cat_id').reset_index(drop=True)  
cat_to_id = dict(cat_id_df.values)  
id_to_cat = dict(cat_id_df[['cat_id', 'cat']].values)  
df.sample(10)  
得到的结果如图 5 所示。
```

	cat	review	cat_id
2914	环境保护	I3县宝塔湖群长出两棵“毒瘤树”	1
8431	卫生计生	有悖道德的地矿医院昧着良心私吞产妇胎盘	6
8166	商贸旅游	兰天东风本田中天店强要订金后不退看车订金，违法乱纪店大欺客	5
2552	环境保护	L3县产大米 多项重金属严重超标	1
5462	劳动和社会保障	L3县合仁坪矿业几十人得矽肺病获赔无门	4
5242	劳动和社会保障	A4区临时环卫工退休工资为何一月才640元？	4
3608	教育文体	麻烦督促下B市教师房屋公积金问题的解决	3
1777	城乡建设	招投标报名费会不会太高了点？	0
1250	城乡建设	L8县县住建局副局长利用职务之便破坏投标程序，干扰营商环境	0
265	城乡建设	A8县28路公交车为何迟迟不更换？	0

	cat	cat_id
0	城乡建设	0
1	环境保护	1
2	交通运输	2
3	教育文体	3
4	劳动和社会保障	4
5	商贸旅游	5
6	卫生计生	6

图 5 转化结果

2.2 分词

1.去停用词

去停用词可以降低句子噪音对句子的理解，减少特征词的数量，从而提高文本分类的准确性。如果英语中 the、is、at、who 等，中文中的在、的、和等副词、量词、介词、叹词、数词等词，这些对理解语句没有实际意义，而且出现频率较高，容易造成噪音，分词后应从分词结果中将这此停用词进行过滤。去停用词只要建立停用词表，然后采用字符匹配的方式扫描分词词典进行删除。去停用词结果如图 6 所示。

#定义删除除字母,数字，汉字以外的所有符号的函数

```
def remove_punctuation(line):
    line = str(line)
    if line.strip() == "":
        return ""
    rule = re.compile(u"^[^a-zA-Z0-9\u4E00-\u9FA5]")
    line = rule.sub("",line)
```

```

return line
def stopwordslist(filepath):
    stopwords = [line.strip() for line in open(filepath, 'r',
encoding='utf-8').readlines()]
    return stopwords
#加载停用词

```

```
stopwords = stopwordslist("C:/chapter12/demo/data/hlt_stop_words.txt")
```

中文停用词包含了很多日常使用频率很高的常用词,如 吧, 吗, 呢, 啥等
一些感叹词等,这些高频常用词无法反应出文本的主要意思,所以要被过滤掉。可
以在这里下载中文停用词。

```
#删除除字母,数字, 汉字以外的所有符号
```

```
df['clean_review'] = df['review'].apply(remove_punctuation)
df.sample(10)
```

	cat	review	cat_id	clean_review
2048	环境保护	箭弓山社区居民的健康谁来保障	1	箭弓山社区居民的健康谁来保障
2553	环境保护	L6县沙溪乡(凤羊)露天石煤矿危害大	1	L6县沙溪乡凤羊露天石煤矿危害大
275	城乡建设	请求延长A市311路公交车至洋湖的报告	0	请求延长A市311路公交车至洋湖的报告
7679	商贸旅游	L5县的乡镇公共汽车什么时候规定婴儿也要收全票	5	L5县的乡镇公共汽车什么时候规定婴儿也要收全票
7965	商贸旅游	A市茂华禧都会B29栋2单元电梯已经坏了将近一个月了	5	A市茂华禧都会B29栋2单元电梯已经坏了将近一个月了
3182	交通运输	对K3县x001公路清江村段路面维修改造的建议	2	对K3县x001公路清江村段路面维修改造的建议
9119	卫生计生	I5县现代医院给婴儿多打了两针疫苗!	6	I5县现代医院给婴儿多打了两针疫苗
2254	环境保护	E1区宝东造纸公司多年来在其生产时晚上不定期偷排污水	1	E1区宝东造纸公司多年来在其生产时晚上不定期偷排污水
3555	交通运输	K4县东站交通和治安问题的建议	2	K4县东站交通和治安问题的建议
2823	环境保护	D1区狮子山私家化工厂排放有毒气体,百姓有怨气	1	D1区狮子山私家化工厂排放有毒气体百姓有怨气

图 6 去除停用词

2.分词

由于中文文本的特点是词与词之间没有明显的界限,从文本中提取词语时需要分词,本文分词采用当前广泛使用的基于 Python 的中文分词工具 jieba, jieba 中文分词工具内置多个算法,支持多种模式进行分词,能有效解决未登陆词和歧义词,准确率高达 97%, 召回率高达 90%。对问题和回答中的每一句话进行分词进行中文分词。jieba 分词用到的算法为最短路径匹配算法该算法首先利用词典找到字符串中所有可能的词条,然后构造一个有向无环图。其中,每个词条对应图中的一条有向边,并可利用统计的方法赋予对应的边长一个权值,然后找到从起点到终点的最短路径,该路径上所包含的词条就是该句子的切分结果。分词结果如 7 所示。

```
df['cut_review'] = df['clean_review'].apply(lambda x: " ".join([w for w
```

```
in list(jb.cut(x)) if w not in stopwords]))
df.head()
```

	cat	review	cat_id	clean_review	cut_review
0	城乡建设	A市西湖建筑集团占道施工有安全隐患	0	A市西湖建筑集团占道施工有安全隐患	A市西湖建筑集团占道施工安全隐患
1	城乡建设	A市在水一方大厦人为烂尾多年，安全隐患严重	0	A市在水一方大厦人为烂尾多年安全隐患严重	A市在水一方大厦人为烂尾多年安全隐患严重
2	城乡建设	投诉A市A1区苑物业违规收停车费	0	投诉A市A1区苑物业违规收停车费	投诉A市A1区苑物业违规收停车费
3	城乡建设	A1区蔡锷南路A2区华庭楼顶水箱长年不洗	0	A1区蔡锷南路A2区华庭楼顶水箱长年不洗	A1区蔡锷南路A2区华庭楼顶水箱长年不洗
4	城乡建设	A1区A2区华庭自来水好大一股霉味	0	A1区A2区华庭自来水好大一股霉味	A1区A2区华庭自来水好大一股霉味

图 7 分词

3.绘制词云图

词云图，也叫文字云，是对文本中出现频率较高的“关键词”予以视觉化的展现，词云图过滤掉大量的低频低质的文本信息，使得浏览者只要一眼扫过文本就可领略文本的主旨。绘制词云图，根据词云图结果来修改停用词的去除，绘制词云图结果如图 8 所示。

```
from collections import Counter
from wordcloud import WordCloud

def generate_wordcloud(tup):
    wordcloud = WordCloud(background_color='white',
                           font_path='simhei.ttf',
                           max_words=50, max_font_size=40,
                           random_state=42
                           ).generate(str(tup))
    return wordcloud

cat_desc = dict()
for cat in cat_id_df.cat.values:

    text = df.loc[df['cat']==cat, 'cut_review']
    text = (' '.join(map(str,text))).split(' ')
    cat_desc[cat]=text

fig,axes = plt.subplots(5, 2, figsize=(30, 38))
k=0
for i in range(5):
    for j in range(2):
        cat = id_to_cat[k]
        most100=Counter(cat_desc[cat]).most_common(100)
        ax = axes[i, j]
        ax.imshow(generate_wordcloud(most100),
```

```

interpolation="bilinear")
ax.axis('off')
ax.set_title("{} Top 100".format(cat), fontsize=30)
k+=1

```



图 8 词云图

2.3 TF-IDF 的特征值

TF-IDF (term frequency-inverse document frequency) 是一种用于信息检索与数据挖掘的常用加权技术。TF 意思是词频(Term Frequency), IDF 意思是逆文本频率指数(Inverse Document Frequency)。TF-IDF 是在单词计数的基础上, 降低了

常用高频词的权重,增加罕见词的权重。因为罕见词更能表达文章的主题思想,比如在一篇文章中出现了“中国”和“卷积神经网络”两个词,那么后者将更能体现文章的主题思想,而前者是常见的高频词,它不能表达文章的主题思想。所以“卷积神经网络”的 TF-IDF 值要高于“中国”的 TF-IDF 值。这里我们会使用 `sklearn.feature_extraction.text.TfidfVectorizer` 方法来抽取文本的 TF-IDF 的特征值。这里我们使用了参数 `gram_range=(1,2)`,这表示我们除了抽取评论中的每个词语外,还要抽取每个词相邻的词并组成一个“词语对”,如: 词 1, 词 2, 词 3, 词 4, (词 1, 词 2), (词 2,词 3), (词 3, 词 4)。这样就扩展了我们特征集的数量,有了丰富的特征集才有可能提高我们分类文本的准确度。参数 `norm='l2'`,是一种数据标准化处理的方式,可以将数据限制在一定的范围内比如说(-1,1)。

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer(norm='l2', gram_range=(1, 2))
features = tfidf.fit_transform(df.cut_review)
labels = df.cat_id
print(features.shape)
print('-----')
print(features)
```

```
(9210, 56980)
-----
(0, 49405) 0.338148392721
(0, 29014) 0.262177633695
(0, 55309) 0.266116526971 (1, 20688) 0.310773761157
(0, 34838) 0.268236330047 (1, 22108) 0.310773761157
(0, 23661) 0.256894651241 (1, 9213) 0.310773761157
(0, 49406) 0.389921452465 (1, 40880) 0.310773761157
(0, 29038) 0.389921452465 (1, 21970) 0.310773761157
(0, 55325) 0.389921452465 (1, 23663) 0.310773761157
(0, 34848) 0.389921452465 (2, 31688) 0.180325425205
(1, 23661) 0.204749229576 (2, 796) 0.235122103997
(1, 20687) 0.310773761157 (2, 15115) 0.349683346383
(1, 22103) 0.24440613452 :
(1, 9210) 0.287942620991 (9207, 49091) 0.388030900982
(1, 40878) 0.269509787601 (9207, 8675) 0.406903927718
(1, 21959) 0.216497521811 (9207, 9090) 0.406903927718
(1, 6940) 0.140937661703 (9208, 3861) 0.234922674424
```

图 9 features 的维度

我们看到我们的 features 的维度是(9210,56980),这里的 9210 表示我们总共有 9210 条留言数据, 56980 表示我们的特征数量这包括全部评论中的所有词语数+词语对(相邻两个单词的组合)的总数。下面我们要是卡方检验的方法来找出每个

分类中关联度最大的两个词语和两个词语对。卡方检验是一种统计学的工具,用来检验数据的拟合度和关联度。在这里我们使用 sklearn 中的 chi2 方法。

```
from sklearn.feature_selection import chi2
import numpy as np

N = 2
for cat, cat_id in sorted(cat_to_id.items()):
    features_chi2 = chi2(features, labels == cat_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(tfidf.get_feature_names())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
    print("# '{}':".format(cat))
    print("    . Most correlated unigrams:\n          . {}".format('\n          . '.join(unigrams[-N:])))
    print("    . Most correlated bigrams:\n          . {}".format('\n          . '.join(bigrams[-N:])))

# '交通运输':
. Most correlated unigrams:
. 的士
. 出租车
. Most correlated bigrams:
. 出租车 管理
. 滴滴 出行
# '劳动和社会保障':
. Most correlated unigrams:
. 职工
. 社保
. Most correlated bigrams:
. 社保 问题
. 退休 人员
# '卫生计生':
. Most correlated unigrams:
. 独生子女
. 医院
. Most correlated bigrams:
. 再婚 家庭
. 人民 医院
# '商贸旅游':
. Most correlated unigrams:
. 电梯
. 传销
. Most correlated bigrams:
. 传销 组织
. 小区 电梯
# '城乡建设':
. Most correlated unigrams:
. 公积金
. 房产证
. Most correlated bigrams:
. 拖欠 工程款
```

图 10 关联度

我们可以看到经过卡方(chi2)检验后，找出了每个分类中关联度最强的两个词和两个词语对。这些词和词语对能很好的反映出分类的主题。

2.4 分类器的选择

为了训练监督学习的分类器，我们首先将“review”转变为包含数字的词向量。例如我们前面已经转换好的 tf-idf 的 features。

当我们有了词向量以后我们就可以开始训练我们的分类器。分类器训练完成后,就可以对没有见过的 review 进行预测。

朴素贝叶斯分类器最适合用于基于词频的高维数据分类器，最典型的应用如垃圾邮件分类器等,准确率可以高达 95% 以上。这里我们使用的是 sklearn 的朴素贝叶斯分类器 MultinomialNB，我们首先将 review 转换成词频向量,然后将词频向量再转换成 TF-IDF 向量，还有一种简化的方式是直接使用 TfidfVectorizer 来生成 TF-IDF 向量(正如前面生成 features 的过程)，这里我们还是按照一般的方式将生成 TF-IDF 向量分成两个步骤：1.生成词频向量. 2.生成 TF-IDF 向量。最后我们开始训练我们的 MultinomialNB 分类器。

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB

X_train, X_test, y_train, y_test = train_test_split(df['cut_review'],
df['cat_id'], random_state = 0)
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

clf = MultinomialNB().fit(X_train_tfidf, y_train)

当模型训练完成后,我们让它预测一些自定义的 review 的分类。不过我们首先编写一个预测函数 myPredict
def myPredict(sec):
    format_sec=" ".join([w for w in
list(jb.cut(remove_punctuation(sec))) if w not in stopwords])
    pred_cat_id=clf.predict(count_vect.transform([format_sec]))
    print(id_to_cat[pred_cat_id[0]])
```

2.5 模型的选择

接下来我们尝试不同的机器学习模型,并评估它们的准确率,我们将使用如下四种模型:

```
Logistic Regression(逻辑回归)
(Multinomial) Naive Bayes(多项式朴素贝叶斯)
Linear Support Vector Machine(线性支持向量机)
Random Forest(随机森林)
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC

from sklearn.model_selection import cross_val_score

models = [
    RandomForestClassifier(n_estimators=200, max_depth=3,
random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
    model_name = model.__class__.__name__
    accuracies = cross_val_score(model, features, labels,
scoring='accuracy', cv=CV)
    for fold_idx, accuracy in enumerate(accuracies):
        entries.append((model_name, fold_idx, accuracy))
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx',
'accuracy'])

import seaborn as sns

sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)

plt.show()
```



图 11 平均准确率

从可以箱体图上可以看出随机森林分类器的准确率是最低的,因为随机森林属于集成分类器(有若干个子分类器组合而成),一般来说集成分类器不适合处理高维数据(如文本数据),因为文本数据有太多的特征值,使得集成分类器难以应付,另外三个分类器的平均准确率都在 80% 以上。其中线性支持向量机的准确率最高。如下所示。

```
cv_df.groupby('model_name').accuracy.mean()

model_name
LinearSVC          0.808468
LogisticRegression 0.718564
MultinomialNB      0.642009
RandomForestClassifier 0.357669
Name: accuracy, dtype: float64
```

2.6 模型的评估

下面我们就针对平均准确率最高的 LinearSVC 模型,我们将查看混淆矩阵,并显示预测标签和实际标签之间的差异。

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
```

1. 训练模型

```
model = LinearSVC()
X_train, X_test, y_train, y_test, indices_train, indices_test =
train_test_split(features, labels, df.index,

test_size=0.33, stratify=labels, random_state=0)
```

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

2.生成混淆矩阵

混淆矩阵的主对角线表示预测正确的数量,除主对角线外其余都是预测错误的数量.从上面的混淆矩阵可以看出"蒙牛"类预测最准确,只有一例预测错误。“平板”和“衣服”预测的错误数量教多。

多分类模型一般不使用准确率(accuracy)来评估模型的质量,因为 accuracy 不能反应出每一个分类的准确性,因为当训练数据不平衡(有的类数据很多,有的类数据很少)时, accuracy 不能反映出模型的实际预测精度,这时候我们就需要借助于 F1 分数、ROC 等指标来评估模型。

```
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,8))
sns.heatmap(conf_mat, annot=True, fmt='d',
             xticklabels=cat_id_df.cat.values,
             yticklabels=cat_id_df.cat.values)
plt.ylabel('实际结果',fontsize=18)
plt.xlabel('预测结果',fontsize=18)
plt.show()
```

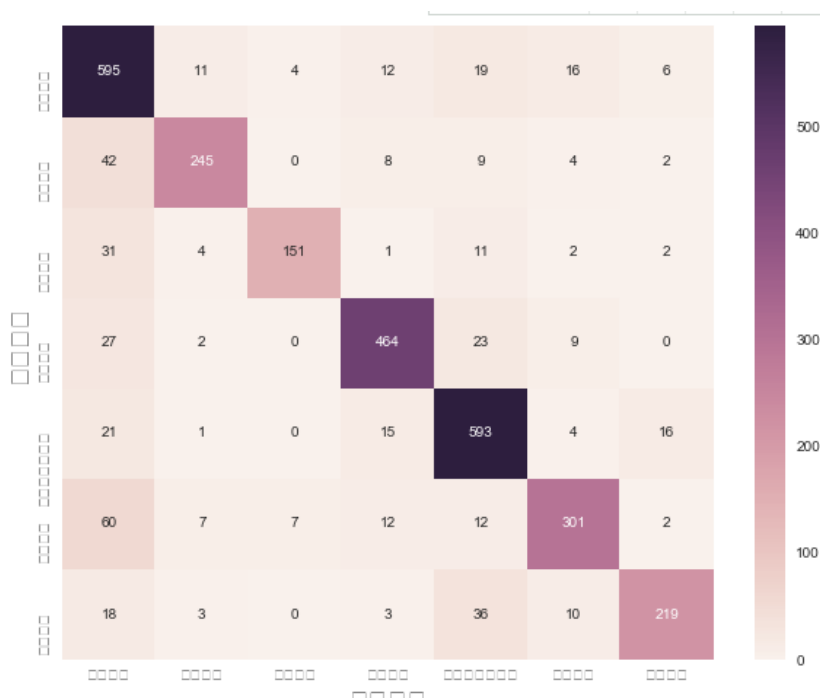


图 12 混淆矩阵

3.查看各个类的 F1 分数

```

from sklearn.metrics import classification_report

print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test,
y_pred,target_names=cat_id_df['cat'].values))

```

```

accuracy 0.844736842105
           precision    recall  f1-score   support

  城乡建设             0.75      0.90      0.82       663
  环境保护             0.90      0.79      0.84       310
  交通运输             0.93      0.75      0.83       202
  教育文体             0.90      0.88      0.89       525
  劳动和社会保障      0.84      0.91      0.88       650
  商贸旅游             0.87      0.75      0.81       401
  卫生计生             0.89      0.76      0.82       289

 avg / total          0.85      0.84      0.84      3040

```

图 13 查看结果

从以上 F1 分数上看,"蒙牛"类的 F1 分数最大(只有一个预测错误),“热水器”类 F1 分数最差只有 66%, 究其原因可能是因为“热水器”分类的训练数据最少只有 574 条,使得模型学习的不够充分,导致预测失误较多吧。

4.查看预测失误的例子,改善分类器

```

from IPython.display import display

for predicted in cat_id_df.cat_id:
    for actual in cat_id_df.cat_id:
        if predicted != actual and conf_mat[actual, predicted] >= 6:
            print("{} 预测为 {}: {} 例.".format(id_to_cat[actual],
id_to_cat[predicted], conf_mat[actual, predicted]))
            display(df.loc[indices_test[(y_test == actual) & (y_pred
== predicted)]][['cat', 'review']])
            print(')

```

环境保护 预测为 城乡建设：42 例。

cat	review
2636 环境保护	L5县福利硅沙厂在我枫香林村轿顶山矿区狂开乱采
2439 环境保护	K8县有很多村子就地焚烧垃圾，往河流乱倒垃圾
2048 环境保护	箭弓山社区居民的健康谁来保障
2423 环境保护	K11县县河路口镇尖山社区白草云村被猪场包围
2040 环境保护	A4区华润万家马路对面饭店油烟乱排影响环境
2467 环境保护	K2区上领桥镇大圪乾村黄泥塘水库承包商用猪粪来养鱼
2071 环境保护	请求依法关停、搬迁B4区江山路侧压缩垃圾中转站
2041 环境保护	在A6区含浦镇开心农场匝道附近天气一放晴就大量烧垃圾

交通运输 预测为 城乡建设：31 例。

cat	review
3301 交通运输	楚江A市段很多渔船停在航道中间影响货船通行
2947 交通运输	请加快A6区时代倾城小区道路交通安全出行措施的建设
3351 交通运输	A3区中国邮政局网点A6区县学士所经常开不了发票
3374 交通运输	国际邮件EMS邮局代收，延迟送货且快件内笔记本电脑主机丢失
3398 交通运输	G4县居民为什么不能办理危险货物运输资格证？

教育文体 预测为 城乡建设：27 例。

cat	review
4070 教育文体	感谢易炼红书记为民解忧！我们的问题解决落实了
4620 教育文体	C市易俗河百花小学老师私自办补习班成群
3709 教育文体	A市理工大学研究生院英语考试出现80%以上的雷同卷
4452 教育文体	A市恒大城幼儿园空置4、5年啦！
4271 教育文体	L5县三杰寄宿学校的教室、宿舍综合楼存在重大安全隐患
4417 教育文体	关于对K6县潇水风光带建设规划设计的建议
3586 教育文体	请解决A4区辰北三角洲片区教育资源不足问题
4819 教育文体	C市“禁机令”目前落实情况如何呢？
4143 教育文体	隐藏于M9县古城的“龙须沟”，脏乱差无人管理

劳动和社会保障 预测为 城乡建设：21 例.

	cat	review
5344	劳动和社会保障	关于A1区两点问题
6503	劳动和社会保障	A2区域南路熙台岭社区为什么要故意设置门槛
6373	劳动和社会保障	K市电业局恒通电力集团的做法请领导管
6394	劳动和社会保障	请问K1区南津渡办事处
5657	劳动和社会保障	D市住房公积金贷款门槛高
5325	劳动和社会保障	请我们政府到J市自来水基层，去深入了解

商贸旅游 预测为 城乡建设：60 例.

	cat	review
8009	商贸旅游	投诉I4县大汉二期翡翠湾房屋质量问题
7874	商贸旅游	咨询申请驰名商标相关资质问题
7139	商贸旅游	举报A2区时代阳光大道99号富景园小区非法传销组织
7742	商贸旅游	L7县苍冲村到兰里街上不到20分钟，经需要4元人民币
8086	商贸旅游	A9市彩虹城房租质量问题急需解决
8043	商贸旅游	A市华嘉地产公司汇金城二期钢化玻璃变普通玻璃

卫生计生 预测为 城乡建设：18 例.

	cat	review
8938	卫生计生	K8县卫生事业只能用脏乱差形容
8746	卫生计生	请西地省卫生计划委员会的领导们处理农卫系统不能正常使用的问题
8889	卫生计生	I2区分局抓走我院领导，医闹何时能停止？
8892	卫生计生	L市上环证如何能辨别真假
8345	卫生计生	A8县药店的GSP认证什么时候有公示？
8604	卫生计生	希望张厅长在百忙之中抽空看看，还K市桥头人民一个公道！
9201	卫生计生	G5县小型餐馆到食品药品监督管理局办卫生许可证要3500多，这合理吗？

图 14 全部查看结果

3. 问题二：热点问题挖掘

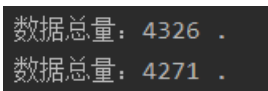
某一时段内群众集中反映的某一问题可称为热点问题，如“XXX 小区多位

业主多次反映入夏以来小区楼下烧烤店深夜经营导致噪音和油烟扰民”。及时发现热点问题，有助于相关部门进行有针对性地处理，提升服务效率。请根据附件 3 将某一时段内反映特定地点或特定人群问题的留言进行归类，定义合理的热度评价指标，并给出评价结果，按表 1 的格式给出排名前 5 的热点问题，并保存为文件“热点问题表.xls”。按表 2 的格式给出相应热点问题对应的留言信息，并保存为“热点问题留言明细表.xls”。

3.1 数据预处理

针对此问题：基本步骤与第一个问题相同，首先先对数据进行简单的清洗，可以先将留言编号和留言用户以及留言时间这三列无用数据删除，然后进行数据的读取与抽取，发现共有 4326 条数据，对数据进行去重，去除完全重复的数据，得到共有 4271 条有效数据，部分代码如下所示，结果如图 15 所示。

```
import pandas as pd
df = pd.read_excel(r'C:\Users\15198\Desktop\附件 3.xlsx')
df = df[['留言主题','留言详情']]
print("数据总量：%d ." % len(df))
df.sample(10)
#去重
df = df[['留言主题','留言详情']].drop_duplicates()
content=df['留言主题']
print("数据总量：%d ." % len(df))
```



```
数据总量: 4326 .
数据总量: 4271 .
```

图 15 数据预处理

3.2 分词

然后开始着手绘制词云图，可以先将刚才的 excel 文件转为文本文件，随后利用 jieba 分词。部分代码如下所示。

```
img = Image.open(path+r'\background.jpg') #打开图片
img_array = np.array(img) #将图片装换为数组
stopword=[''] #设置停止词
wc = WordCloud(
    background_color='white',
    width=1000,
    height=800,
    mask=img_array, #设置背景图片
```



```
font_path=font,
stopwords=stopword
)
```

利用 matplotlib 和 wordcloud 绘制词云图。如图 16 所示。



图 16 词云图

由词云图可以看出，该市的主要问题是小区业主反映的问题，很有可能是与开发商的问题没有得到解决，然后希望相关部门能够出面管理，还有就是学校部分可能也有问题需要解决，所以热点问题就应该围绕小区周围的问题来找，发现很多小区居民都反映小区周围新建的搅拌站污染环境严重扰民等。

3.3 基于相似度的自适应最优 LDA 模型

1. LDA (Latent Dirichlet Allocation) 简介

LDA (Latent Dirichlet Allocation) 是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。LDA 是一种非监督机器学习技术，可以用来识别大规模文档集 (document collection) 或语料库 (corpus) 中潜藏的主题信息。它采用了词袋 (bag of words) 的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

对于语料库中的每篇文档，LDA 定义了如下生成过程 (generative process)：

- (1) 对每一篇文档，从主题分布中抽取一个主题；
- (2) 从上述被抽到的主题所对应的单词分布中抽取一个单词；

(3) 重复上述过程直至遍历文档中的每一个单词。

语料库中的每一篇文档与 T (通过反复试验等方法事先给定) 个主题的一个多项分布 (multinomial distribution) 相对应, 将该多项分布记为 θ 。每个主题又与词汇表 (vocabulary) 中的 V 个单词的一个多项分布相对应, 将这个多项分布记为 β 。如图 17 所示。

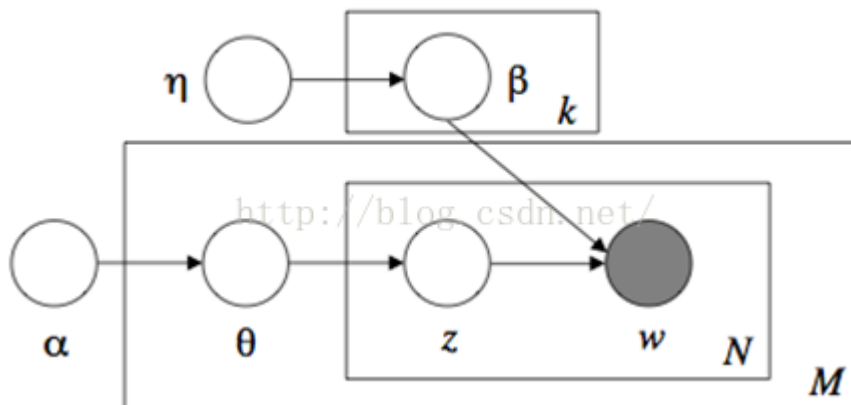


图 17 LDA 的图模型表示

2. 基于相似度的自适应最优 LDA 模型

采用余弦相似度来度量文本的相似度, 该公式如下:

$$\text{sim}(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\| \|d_j\|}$$

表示两个向量的内积, 即假设两个向量为 n 维向量, 则

$$d_i \cdot d_j = \sum_{k=1}^n w_{i,k} \cdot w_{j,k}$$

可以通过下式计算:

$$\|d_i\| \|d_j\| = \sqrt{w_{i,1}^2 + w_{i,2}^2 \dots w_{i,n}^2} \sqrt{w_{j,1}^2 + w_{j,2}^2 \dots w_{j,n}^2}$$

现在可以通过余弦相似度公式计算上述两个文档的相似度。

$$\text{sim}(d_1, d_2) = \frac{w_{1,1} \cdot w_{2,1} + w_{1,2} \cdot w_{2,2} \dots w_{1,23} \cdot w_{2,23}}{\sqrt{w_{1,1}^2 + w_{1,2}^2 \dots w_{1,23}^2} \cdot \sqrt{w_{2,1}^2 + w_{2,2}^2 \dots w_{2,23}^2}} = \frac{1}{\sqrt{12} \cdot \sqrt{12}}$$

最后我们就需要确定阈值来进行判断两个文档是否相似。

3.python 实现

```

# 构造主题数寻优函数
def cos(vector1, vector2): # 余弦相似度函数
    dot_product = 0.0;
    normA = 0.0;
    normB = 0.0;
    for a,b in zip(vector1, vector2):
        dot_product += a*b
        normA += a**2
        normB += b**2
    if normA == 0.0 or normB==0.0:
        return(None)
    else:
        return(dot_product / ((normA*normB)**0.5))

# 主题数寻优
def lda_k(x_corpus, x_dict):
    # 初始化平均余弦相似度
    mean_similarity = []
    mean_similarity.append(1)
    # 循环生成主题并计算主题间相似度
    for i in np.arange(2,11):
        lda = models.LdaModel(x_corpus, num_topics = i, id2word = x_dict)

# LDA 模型训练
        for j in np.arange(i):
            term = lda.show_topics(num_words = 50)

            # 提取各主题词
            top_word = []
            for k in np.arange(i):
                top_word.append([".".join(re.findall("(.*)",i)) for i in

```

```

term[k][1].split('+')) # 列出所有词

# 构造词频向量
word = sum(top_word,[]) # 列出所有的词
unique_word = set(word) # 去除重复的词

# 构造主题词列表，行表示主题号，列表示各主题词
mat = []
for j in np.arange(i):
    top_w = top_word[j]
    mat.append(tuple([top_w.count(k) for k in unique_word]))

p = list(itertools.permutations(list(np.arange(i)),2))
l = len(p)
top_similarity = [0]
for w in np.arange(l):
    vector1 = mat[p[w][0]]
    vector2 = mat[p[w][1]]
    top_similarity.append(cos(vector1, vector2))

# 计算平均余弦相似度
mean_similarity.append(sum(top_similarity)/l)

return(mean_similarity)

```

通过分析，排名前五的问题如下图 18 所示。

热度排名	问题ID	热度指数	时间范围	地点/人群	问题描述
1	1	45	2019/11/2至2020/1/26	A2区丽发新城	小区旁建搅拌站，污染环境，严重扰民
2	2	11	2019/1/6至2019/5/22	A市辉煌国际城	居民楼下开饭店非法营业
3	3	11	2019/1/6至2019/9/12	A3区西湖街道茶场村五组	什么时候能启动征地拆迁，拆迁规划是什么
4	4	10	2019/7/21至2019/09/25	A5区劳动东路魅力之城小区	小区临街烧烤摊油烟噪音扰民
5	5	9	2019/1/3至2019/12/30	A市地铁三号线	地铁三号线出入口设置不合理

图 18 排名前五的问题

找到热点问题后，热度指数可以用该问题出现的次数来评价，出现次数越多热度指数越多。然后根据热点问题找到对应详细的留言明细，保存为“热点问题

留言明细表.xls”。如图 19 所示。

问题ID	留言编号	留言用户	留言主题	留言时间	留言详情	反对数	点赞数
1	188809	A909139	丽南路丽发新城居民区附近搅拌站	2019/11/19 18:07:54	旁50米处建搅拌站，运渣车吵	0	1
1	189950	A909204	A2区丽发新城附近建搅拌站噪音	2019-11-13 11:20:21	的地方建搅拌站。可想而知，	0	0
1	190108	A909240	丽发新城小区旁边建搅拌站	2019-12-21 15:11:29	严重影响几千名学生的健康。	0	1
1	190523	A00072847	新城违建搅拌站，彻夜施工扰民	2019/12/26 13:55:15	严重；3、搅拌站几百米外新	0	0
1	199379	A00092242	发新城附近修建搅拌厂，严重污	2019/11/25 10:17:56	民因此还得了疾病住院，该地	0	0
1	203393	A00053065	区侧面建设混凝土搅拌站，粉尘	2019/11/19 14:51:53	巨大的粉尘，严重影响居民	0	2
1	208714	A00042015	新城附近修建搅拌站，污染环境	2020-01-02 00:00:00	质量和声环境质量急剧下降	0	4
1	213464	A909233	发新城小区附近违建搅拌站噪音	2019-12-10 12:34:21	型搅拌站。该搅拌站的设备	0	0
1	213930	A909218	新城附近违规乱建混凝土搅拌站	2019-12-27 23:34:32	居民强烈呼吁政府和有关职能	0	0
1	214282	A909209	新城小区附近搅拌站噪音扰民和	2020-01-25 09:07:21	天天吵，烦死了不仅吵还臭	0	0
1	215563	A909231	发新城小区旁边的搅拌厂是否合	2019-12-06 12:21:32	产生了噪音和灰尘。这给小区	0	0
1	215842	A909210	A2区丽发新城小区附近太吵了	2020-01-26 19:47:11	一个搅拌厂是怎么回事！下班	0	0
1	216824	A909214	加工砂石料噪音污水影响丽发新	2019-12-25 12:15:57	区，这些严重扰民的噪音是从	0	0
1	217700	A909239	发新城小区旁的搅拌站严重影响	2019-12-21 02:33:21	区内搬迁到丽发新城小区旁	0	1
1	222831	A909228	污染的A2区丽发新城附近环保	2019-12-22 10:23:11	城里能修改产生大量灰尘的	0	0
1	225217	A909223	发新城附近修建搅拌厂严重影响	2019-11-15 09:17:36	新建一搅拌站，每天尘土飞扬	0	0
1	231136	A909204	A2区丽发新城附近建搅拌站噪音	2019-12-02 11:20:21	站。距离上次投诉已经过去一	0	0
1	233158	A909242	发新城小区旁建搅拌厂严重扰民	2019-12-05 08:46:20	班不在家我还能忍，但周末	0	0
1	235362	A909215	发新城小区附近水泥搅拌站非法	2020-01-06 20:45:34	肆虐，严重危害居民身体健康	0	0
1	238212	A909203	发新城小区附近建搅拌站合理吗	2019-12-12 10:23:11	区作为居民区应是一个安静的	0	0

图 19 热点问题明细表部分截图

4. 问题三：答复意见的评价

首先由答复时间和提问时间可以看出答复的时效性并不高，经常要半个月到一个月才能得到答复，让问题一拖再拖无法得到及时的解决。

答复的相关性良好，都会针对市民提出的问题进行答复。可解释性较强，都会根据相应问题来提出相应的举措，或者告诉市民如何能够更有效的解决问题。完整性略有欠缺，并不能全面完整的回应市民提出的所有问题，部分问题还是选择了忽略回避，这点仍需加强。

对于市民提出的问题要逐一进行答复，答复意见要明确具体。用词造句要通俗易懂，表述准确，简明扼要，不能避而不答，且最好早点给予答复。

针对答复内容的评价，本文采取相关性进行评价，基于第一题的模型建立，已经对每种类型的群众留言已经做了数据处理，政府的回答肯定是和群众留言内容相关，答复内容就和相应的群众留言类别具有相关性。为了简化数据，本文训练数据选择 label1。

4.1 模型训练

根据第一题，同理可得相应数据模型。

1.导入自定义的分词词典

```
import jieba

import jieba.posseg as pseg

import jieba.analyse

import pymssql
```

```
import xlwt

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer,
TfidfTransformer

import pandas as pd

jieba.load_userdict('newdic1.txt')
```

2. 连接数据库

```
conn = pymysql.connect(host='1.1.1.1', user='username', password='password',
database='database', charset='utf8')

cur = conn.cursor()

sql = 'select distinct(column) from table'

cur.execute(sql)

list1 = cur.fetchall()
```

3. 分词

```
words = [ ]

for word in list1:
    for i in word:
        seg_list = jieba.cut(i, cut_all=False)
        words.append(" ".join(seg_list))
```

4. 计算 tfidf

```
tfidf_vectorizer = CountVectorizer(min_df=2, max_df=0.8,
token_pattern=u'(?u)\b[^\d\W]+\b')

transformer = TfidfTransformer()

word_tfidf = tfidf_vectorizer.fit_transform(words)

tfidf = transformer.fit_transform(word_tfidf)

print(tfidf.shape)
```

5. 获取特征词

```
features = tfidf_vectorizer.get_feature_names()
```

6. 加载已训练好的 word2vec 模型

word2vec 是轻量级的神经网络，其模型仅仅包括输入层、隐藏层和输出层，模型框架根据输入输出的不同，主要包括 CBOW 和 skip-gram 模型，CBOW 模型是通过上下文的内容预测中间的目标词，而 skip-gram 则相反，通过目标词预测其上下文的词，通过最大化词出现的概率，我们训练模型可得到各个层之间的权重矩阵，我们所说的得到的 word embedding vector 就是从这个权重矩阵里面得来的。

```
from gensim.models import word2vec
import logging
logging.basicConfig(format='%(asctime)s: %(levelname)s: %(message)s',
                    level=logging.INFO)
model = gensim.models.KeyedVectors.load_word2vec_format('model_word.bin',
                binary=True)
#建立 word2vec 模型
import jieba
fin=open('message_label.csv','r',encoding='gb18030')
fou=open('newdic1.txt','w',encoding='utf-8', errors='ignore')
line =fin.readline()
while line:
    newline=jieba.cut(line,cut_all=False)
    str_out=' '.join(newline).replace(',','').replace('。',
    ',').replace('?', '').replace('!', '') \
        .replace('"', '').replace("'", '').replace(':',
    ',').replace('“', '').replace('”', '').replace('—', '') \
        .replace('（', '').replace('）', '').replace('《',
    '》').replace('；', '').replace('.', '') \
        .replace('&#39;',
    '').replace('...', '').replace('&#39;',
    ',').replace('!', '')
    print(str_out, file=fou)
    line=fin.readline()
```

```

fin.close()
fou.close()

import gensim.models.word2vec as w2v
model_file_name = 'message -label1.csv'
sentences = w2v.LineSentence('message -label1.csv')
model = w2v.Word2Vec(sentences, size=20, window=5,
min_count=5, workers=4)
model.save(model_file_name)

```

7. 打开要匹配相似度的文本

```

f4 = open('留言详情.xlsx', encoding='utf-8', errors='ignore')

ff = []

for j in f4.readlines():

    j = j.replace('\n', '')

    ff.append(j)

print(len(ff))

```

8. tfidf 稀疏矩阵

```

tfidf_tuple = tfidf.nonzero()

tfidf_rows = tfidf_tuple[0]

tfidf_columns = tfidf_tuple[1]

size = len(tfidf_columns)

print('nonzero.size=%s' % size)

```

9.将 TF-IDF>=某一阈值 d 的数据存入字典

```

product_dict = {}

for i in range(size):

    row = tfidf_rows[i]

    column = tfidf_columns[i]

    tfidf_value = tfidf[row, column]

    if tfidf_value <= 0.4:

        continue

    key_words = product_dict.setdefault(row, [])

```



```

key_word = {}

key_word["key_word"] = features[column]

key_word["tfidf_value"] = tfidf_value

key_words.append(key_word)

print('product_dict.len=%s' % len(product_dict))

```

4.2 相似度计算

```

f = open('gabbage.txt', 'w', encoding='utf-8', errors='ignore')

word2vec = {}

for i in range(size):

    column = tfidf_columns[i]

    cate = word2vec.setdefault(features[column], [])

    for jj in ff:

        try:

            y1 = model.wv.similarity(features[column], jj)

            insert = {"category_name": jj, "similarity": y1}

            cate.append(insert)

        except:

            f.write('')

print('word2vec.len=%s' % len(word2vec))

f.close()

```

1. 相似度值排序

```

import operator

for k, v in word2vec.items():

    new_dict_list = []

    s = sorted(v, key=operator.itemgetter("similarity"), reverse=True)

    word2vec[k] = s

    for w in word2vec[k]:

        if w not in new_dict_list:

            new_dict_list.append(w)

```

```
word2vec[k] = new_dict_list
```

2. 创建满足某一阈值的 tfidf 的特征词的相似度的词典

```
l = [l1 for l1 in product_dict.keys() for l1 in product_dict[l1]]  
  
for k in word2vec.keys():  
  
    for m in l:  
  
        if m["key_word"] == k:  
  
            m["category_names"] = word2vec[k]
```

```
print(len(product_dict))
```

3. 将字典转为数组，索引转为留言主题

```
result = [r for r in list1]  
  
product_list = []  
  
for i in product_dict:  
  
    product = {}  
  
    product['product'] = result[i][0]  
  
    product['key_words'] = product_dict[i]  
  
    product_list.append(product)  
  
print('product_list.len=%s' % len(product_list))  
  
print(len(product_list))
```

4. 匹配填充 tfidf 值满足某一阈值的相似度词典

```
f = open('gabbage.txt', 'w', encoding='utf-8', errors='ignore')
```

4.3 提取特征

将 TF-IDF \geq 0.4 的数据存入字典

```
product_dict = {}  
  
for i in range(size):  
  
    row = tfidf_rows[i]  
  
    column = tfidf_columns[i]  
  
    tfidf_value = tfidf[row, column]  
  
    if tfidf_value <= 0.4:  
  
        continue  
  
    key_words = product_dict.setdefault(row, [])
```

```

for jj in ff:
    try:
        y1 = model.wv.similarity(features[column], jj)
        insert = {'key_word': features[column], 'tfidf_value': tfidf_value,
                  'category_names': {'goods_name': jj, 'word2vec': y1}}
        key_words.append(insert)
    except:
        f.write('')
f.close()

```

4.4 提取结果

```

import json
t = open('word2vec_result1.txt', 'w', encoding='utf-8', errors='ignore')
json_string =
json.dumps(product_list[0:11]).encode('utf-8').decode('unicode_escape')
print(json_string)
t.write(json_string)
t.close()

```

5. 结论

总结本次比赛，我们采用当前广泛使用的基于 Python 的中文分词工具 jieba 进行文本的分词和一系列数据处理的操作，利用 sklearn 实现多分类 demo，从而将群众留言进行分类。再用类似于群众留言分类的方式进行热点问题的挖掘，最后对答复意见进行相应的评价。

参考文献

- [1] <https://blog.csdn.net/xyz1584172808/java/article/details/82353575>
- [2] 胡小娟 基于特征选择的文本分类方法研究 [D]. （吉林大学计算机软件与理论,2018）
- [3] <https://blog.csdn.net/hellozhxy/article/details/82083226>

[4] https://blog.csdn.net/qq_17854471/article/details/104043071?ops_request_misc=%257B%2522request%255Fid%2522%253A

[5] https://blog.csdn.net/weixin_42608414/article/details/88046380?utm_medium=distribute.pc_relevant.none-task-blog-baidujs-1