“智慧政务”中的数据挖掘与综合分析

摘要

互联网的迅速发展使各类网络问政平台成为政府了解民意的重要渠道，随之而来的便是庞大的民意相关文本数据，这给相关部门的工作带来了前所未有的挑战。本文运用了多种数据挖掘方法来解决上述问题，并对相应的挖掘方法及效果进行了综合评价。

对于问题一，首先采用 Python 中的中文分词包 jieba 进行信息分词，以及利用 TF-IDF 算法将文档相似度的问题转换为数学向量矩阵问题，通过 VSM 向量空间模型来存储每个文档的词频和权重，根据每条留言内容的 TF-IDF 权重向量，采用朴素贝叶斯分类器对留言进行分类，最终通过分类模型得到所需要的一级标签分类，并对分类结果进行 F-score 评价。

对于问题二，首先先对原始数据进行初步筛选，然后利用 jieba 分词对提取到的留言进行分词处理，然后利用 Python 中的第三方库 sklearn 实现 TF-IDF 算法对数据进行向量化，并提取文本关键字，之后运用 PCA 降维方法将数据降为二维数据，再利用 K-Means 文本聚类方法，对留言进行归类，最后设置合适的热点评价标准对分类好的数据进行分类排序得到结果。

对于问题三，我们分别就答复的相关性、完整性、可解释性等角度分析了数据，在对数据预处理时先判断字符串中的字符是否为汉字，而对于字符串中的数字和标点不做处理，然后对获取的两个字符串进行 Cosine Similarity 的计算，再将文本映射到向量空间，形成中文文本和向量之间的映射关系，从而计算出几个或者多个不同向量的差异大小，得到相应的 Cosine Similarity 值。再通过对答复意见的 Cosine Similarity 值和时间差的计算得到答复意见质量指数，生成了答复意见质量指数表，并对答复意见总体质量进行了相关的分析和评价。

关键词：jieba 中文分词　TF-IDF 算法　朴素贝叶斯分类器　K-means 文本聚类　Cosine Similarity 值

Data Mining and Comprehensive Analysis in "Intelligent Government Affairs"

Abstract

With the rapid development of the Internet, all kinds of online political platforms become an important channel for the government to understand public opinions, which is followed by a huge amount of text data related to public opinions, which brings unprecedented challenges to the work of relevant departments. In this paper, a variety of data mining methods are used to solve the above problems, and the corresponding mining methods and effects are comprehensively evaluated.

For the first problem, first used in Python in Chinese word segmentation packages jieba participle of information, and the use of TF - IDF algorithm converts the problem of document similarity to mathematical vector matrix problems, through the VSM vector space model to store each document's word frequency and weight, according to every message content of TF - IDF weight vector, using naive bayesian classifier to classify messages, finally the level needed to label classification is obtained by classification model, and the classification results are F - score evaluation.

For the second problem, the first to preliminary screening of original data, then use jieba participle word processing to extract the message, and then use a third-party library in Python sklearn implementation TF - IDF vectorization algorithm for data, and extract the text key words, after using PCA dimension reduction method will be reduced to 2D data, using the K - Means text clustering method, to classify the message finally setting up the appropriate evaluation criterion for classification of the hot spot of good data to sort results.

For the third problem, we will reply the relevance, integrity and interpretability of Angle, analyzes the data in the data preprocessing to judge whether the characters in a string of characters, and for a string of Numbers and punctuation don't do processing, and then to obtain the two strings of Cosine Similarity calculation, and then map the text to vector space, form the mapping relationship between Chinese text and vector,

and a few or many different vector is calculated difference size, get the corresponding Cosine Similarity values. Then, by calculating the Cosine Similarity value and time difference of the response, the quality index of the response was obtained, the quality index table of the response was generated, and the overall quality of the response was analyzed and evaluated.

Key words: jieba Chinese word segmentation TF-IDF algorithm Naive Bayesian Classifier K-means text clustering Cosine Similarity value

目录

1 绪论

1.1 挖掘背景

伴随着互联网科技的飞速发展，网络沟通民意的各种方式和机制得到不断地健全和完善。互联网成为了政府同群众交流和了解群众的新平台。微信和微博等各大社交平台以及阳光热线等网络问政平台逐渐成为政府了解民意的重要渠道，越来越多人民群众在各渠道上吐露心声，各类社情民意相关的文本数据量不断攀升，数据量的急剧增长也给相关部门的工作带来了巨大的挑战，然而以往这些数据主要通过人工来对留言进行划分以及对相关热点问题的整理。

在处理网络问政平台的群众留言划分上，工作人员会首先按照一定的划分体系对留言进行划分，以便于后续将群众留言分派至相应的职能里。而目前大部分的电子政务系统依旧是依靠人工根据经验来处理的，存在工作量大、效率低、出错率高等问题，而利用数据的文本分析与挖掘技术可以更加直观地看到数据内容，针对性强且耗时短，在极大程度上提高了工作效率。

1.2 挖掘目标

本次建模目标是对互联网群众的问政留言记录以及相关部门对部分群众的答复数据，利用 jieba 中文分词、TF-IDF 算法、朴素贝叶斯分类和 K-means 聚类方法达到以下三个目标：

① 利用中文分词和分类方法建立关于留言内容的一级标签分类模型，并使用 F-score 对分类方法进行评价；

② 对群众反映的问题留言进行归类分析，并结合合理的热度评价指标进行排序；

③ 针对相关部门对留言的答复意见，从答复的相关性、完整性、可解释性等角度对答复意见的质量给出评价方案。

2 问题 1 分析方法与过程

2.1 操作流程图

读取数据，对数据进行清洗采用 jieba 进行中文分词，划分训练集和测试集，利用朴素贝叶斯分类模型进行分类，最后采用 F-score 评价模型进行评价。操作流程图如图 2.1 所示：

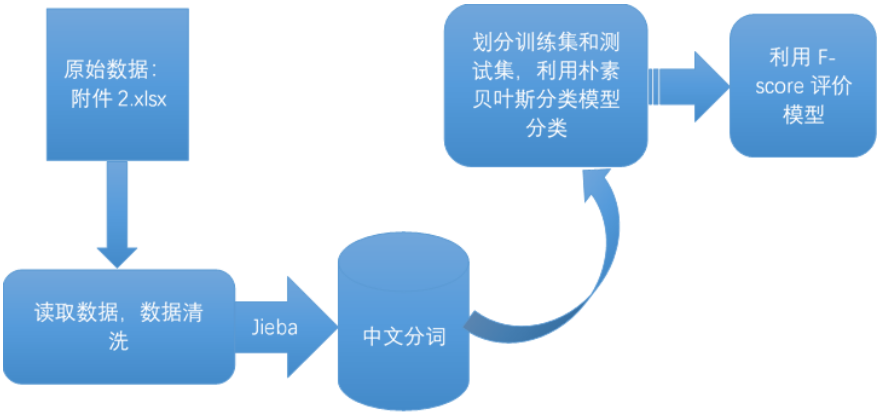

图 2.1 问题 1 流程图

2.2 数据预处理

2.2.1 读取数据及中文分词

采用python中的xlrd包读取附件2中的数据，在进行数据挖掘与分析之前，需要将自然语言的文本信息转换为计算机所能识别的结构化信息，第一步即是将所有留言信息进行中文分词，我们采用python中的中文分词包jieba进行信息分词。jieba分词器是基于python语言的开源库，它自带一个包含 2 万多条中文词的词典，该词典保存在 dict.txt 文件中。jieba分词还提供了精确模式、全模式、搜索引擎模式三种工作模式。精确分词模式由于其总试图将文本以最精确的方式进行词分割，所以比较适合在文本分析领域中使用。jieba分词切换到全模式下时，分词器能把文本中所有能构成词的词汇都找出来，但不能很好的解决词汇歧义现象。搜索引擎模式是精确模式的一种扩展，是对分割后的长词进行二次分割，该模式下可以获得更高的召回率。在分词的同时，采用了TF-

IDF算法，抽取每个留言内容中的前5个关键词，这里采用jieba自带的语义库。

2.2.2 TF-IDF 算法

需要将文档相似度问题转换为数学向量矩阵问题，可以通过VSM向量空间模型来存储每个文档的词频和权重，特征抽取完后，因为每个词语对实体的贡献度不同，所以需要对这些词语赋予不同的权重。计算词项在向量中的权重方法——TF-IDF。

TF-IDF简言之，就是对一篇文章提取关键词的算法。不论怎么将其复杂化定义，其本质都是计算特征词在文章中的重要程度。

第一步：考察一个词的重要程度，最简单的思路就是考察起出现的频率，即TF（Term Frequency）词频。可以通过统计特征词在文章中出现的次数来计算。

$$\mathrm{TF}=\frac{\text{特征词在文章中出现的次数}}{\text{文章中的总次数}} \tag{1}$$

第二步：过滤“停用词”。显然第一步有一个漏洞，比如类似“的”，“是”，“因此”等词语在一篇文章中很常用，我们称其为“停用词”，那么必须考虑过滤掉这些词语，这些词语可以通过一个停用词库手动过滤。

$$IDF=\log\left(\frac{\text{语料库文档总数}}{\text{包含该词的文档数}+1}\right) \tag{2}$$

第三步：计算逆文档频率。过滤掉“停用词”后，只考虑有意义的词，这就引入了另外一个思想IDF（Inverse Document Frequency）逆文档频率，其思想是“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。”

第四步：计算TF-IDF值。将TF和IDF相乘结合后，就可以得到一个特征词的TF-IDF值，作为重要程度的参考。

$$TF-IDF=TF*IDF \tag{3}$$

实际分析得出TF-IDF值与一个词在留言信息表中一句文本出现的次数成正比，某个词文本的重要性越高，TF-IDF值越大。计算文本中每个词的TF-IDF

值，进行排序，次数最多的即为要提取的留言信息表中一句文本的关键词。

2.3 朴素贝叶斯分类器[1]

在贝叶斯分类模型中，朴素贝叶斯分类模型是最简单且有效的分类模型，它采用了属性条件独立性假设，即对已知类别，假设所有属性相互独立，其理论基础是贝叶斯定理[2]。

给定由d个属性描述的一个训练样本集数据$X = x_1, x_2, x_3, \dots, x_d$, 其中$x_i$表示$X$在第$i$个属性上的取值, 有$m$种可能的类别标记, 即$C = C_1, C_2, \dots, C_m$。朴素贝叶斯分类器将未知类别的样本$X$分配给类别$C_k$的条件是$P(C_k|X) > P(C_j|X), 1 \leq k \leq \mathrm{m}, 1 \leq \mathrm{j} \leq \mathrm{m}, \mathrm{j} \neq \mathrm{k}$。根据贝叶斯定理可得：

$$P(C_k|X) = \frac{P(C_k)P(X|C_k)}{P(X)} = \frac{P(C_k)}{P(X)}\prod_{i=1}^{d} P(x_i|C_k) \tag{4}$$

对于所有的类，若$P(X)$相同，则朴素贝叶斯分类模型为：

$$h_n b(X) = \max_{C_k \in C} P(C_k)\prod_{i=1}^{d} P(x_i|C_k) \tag{5}$$

由此可知，朴素贝叶斯分类器的训练过程是基于训练样本集D来估计类先验概率$P(C_k)$，并为每个属性估计条件概率$P(x_i|C_k)$。令D_{C_k}表示训练集D中第C_k类样本集合，则：

$$P(C_k) = \frac{|D_{C_k}|}{|D|} \tag{6}$$

对于离散属性，令D_{C_k,x_i}表示D_{C_k}中第i个属性上取值为x_i的集合，则：

$$P(x_i|C_k) = \frac{|D_{C_k,x_i}|}{|D_{C_k}|} \tag{7}$$

对于连续属性，假设$P(x_i|C_k) \sim N(\mu_{C_k,i}, \sigma^2_{C_k,i})$，其中$\mu_{C_k,i}$和$\sigma^2_{C_k,i}$分别是第$C_k$类样本在第$i$个属性上取值的均值和方差，则：

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi}\sigma_{C_k,i}} \exp\left(-\frac{(x_i - \mu_{C_k,i})^2}{2\sigma^2_{C_k,i}}\right) \tag{8}$$

需要注意，在上述模型中如果某个属性值在训练集中没有某个类别同时出

现，则计算出的概率为零，从而出现误差。利用拉普拉斯修正算法可以解决上述问题：

$$P(C_k)=\frac{|D_{C_k}|+1}{|D|+N},P(x_i|C_k)=\frac{|D_{C_k,x_i}|+1}{|D_{C_k}|+N_i} \tag{9}$$

其中，N表示训练样本集的所有可能类别数，N_i表示第i个属性可能的取值数。

我们这里通过Python中sklearn自带的朴素贝叶斯分类器对留言进行一级标签分类，将训练集和测试集按照7:3的比例随机分离，并得出结果。

2.4 评价分类方法

F-measure是一种统计量,F-Measure又称为F-Score,F-Measure是Precision和Recall加权调和平均，是IR（信息检索）领域的常用的一个评价标准，常用于评价分类模型的好坏[3]。

1. **准确率（accuracy）**，即所有的预测正确（正类负类）的占总的比重。

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN} \tag{10}$$

2. **精确率（也叫查准率，precision)**，即正确预测为正的占全部预测为正的比例（真正正确的占所有预测为正的比例）：

$$Precision=\frac{TP}{TP+FP} \tag{11}$$

3. **召回率（recall）**，即正确预测为正的占全部实际为正的比例（真正正确的占所有实际为正的比例）：

$$Recall=\frac{TP}{TP+FN} \tag{12}$$

4. **F-score值**，F1值为算数平均数除以几何平均数，且越大越好，将Precision和Recall的上述公式带入会发现，当F1值小时，True Positive相对增加，而false相对减少，即Precision和Recall都相对增加，即F1对Precision和Recall都进行了加权，得：

$$\frac{2}{F_1}=\frac{1}{Precision}+\frac{1}{Recall}\ , \tag{13}$$

公式转化为：

$$F_1=\frac{2PR}{P+R}=\frac{2TP}{2TP+FP+FN}\ , \tag{14}$$

在本题中，F-score计算公式为：

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i + R_i} \tag{15}$$

其中P_i为第i类的查准率， R_i为第i类的查全率。

准确率和召回率是互相影响的，理想情况下肯定是做到两者都高，但是一般情况下准确率高、召回率就低，召回率低、准确率高。依题意，我们采用提高准确率的方式来建立模型对附件2给出的数据用朴素贝叶斯分类器进行分类，建立关于留言内容的一级标签分类模型，最后进行模型评价。

2.5 问题 1 结果分析

我们通过朴素贝叶斯分类建立关于留言内容的一级标签分类模型，使用 F-Score 对分类方法进行评价。我们将全部数据划分为训练集和测试集，其中训练集的整体准确率为99.6%，测试集的整体准确率为99.7%（具体实现程序见附件程序：main.py）。

训练集的F-Score评价整体结果如表2.1所示：

表2.1训练集详细评价表

	precision	recall	f1-score	support
0	1.00	0.98	0.99	437
1	0.99	1.00	0.99	1355
2	1.00	0.99	1.00	630
3	1.00	1.00	1.00	853
4	0.99	1.00	1.00	1417
5	1.00	1.00	1.00	1107
6	1.00	1.00	1.00	648
accuracy			1.00	6447
macro avg	1.00	0.99	1.00	6447
weighted avg	1.00	1.00	1.00	6447

从表中我们可以清晰的得出我们各类的准确率都非常的高，为了能够更加

清晰的显示出我们分类的结果，我们出输出了分类结果的混淆矩阵，训练集的混淆矩阵结果如图2.2所示，其中A-G和0-6分别代表：城乡建设，环境保护，交通运输，教育文体，劳动和社会保障，商贸旅游，卫生计生。

输出混淆矩阵：

	A	B	C	D	E	F	G
0	428	5	0	0	4	0	0
1	0	1354	0	1	0	0	0
2	0	1	624	0	5	0	0
3	1	1	0	851	0	0	0
4	0	0	0	0	1417	0	0
5	0	3	0	0	2	1102	0
6	0	3	0	0	0	0	645

图2.2训练集混淆矩阵

由图我们可以得出只有极少数留言被分错，例如本应该为商贸旅游的3条留言被错分为环境保护以及本应该为城乡建设的被错分为劳动与社会保障等。

测试集的F-Score评价整体结果如表2.2所示：

表2.2测试集详细评价表

	precision	recall	f1-score	support
0	0.99	0.99	0.99	176
1	0.99	1.00	1.00	614
2	1.00	0.99	1.00	247
3	1.00	0.99	0.99	362
4	1.00	1.00	1.00	592
5	1.00	1.00	1.00	482
6	1.00	1.00	1.00	290
accuracy			1.00	2763
macro avg	1.00	1.00	1.00	2763
weighted avg	1.00	1.00	1.00	2763

从表中我们可以看出，相比于训练集，测试集的准确率更高，这说明我们的分类模型拟合的非常好，并没有出现过拟合或者欠拟合的问题。同样我们生产测试集的混淆矩阵，如图2.3所示。

输出混淆矩阵：

	A	B	C	D	E	F	G
0	174	1	0	0	1	0	0
1	0	613	0	0	1	0	0
2	0	1	245	1	0	0	0
3	1	2	0	359	0	0	0
4	0	0	0	0	592	0	0
5	0	1	0	0	0	481	0
6	0	0	0	0	0	0	290

图2.3测试集混淆矩阵

混淆矩阵更加清晰的显示了我们分类模型在分类过程的错分的具体类别，从混淆矩阵的具体数据中我们也可以感受到朴素贝叶斯分类器的高准确率。

3 问题 2 分析方法与过程

3.1 问题 2 流程图

通过对问题二的具体分析，问题二要求我们对附件 3 中的数据做舆情分析的处理，舆情分析首先要对数据进行处理，再做自然语言分析，最后提取热点留言并制定合理的热度评价指标。通过分析得到问题二的流程图如图 3.1 所示。

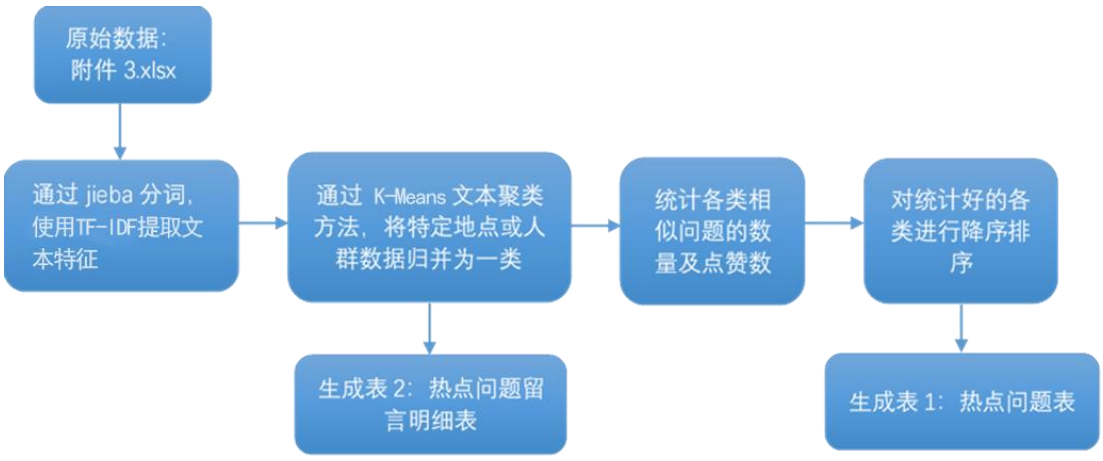

图 3.1 问题 2 流程图

3.2 数据预处理

3.2.1 数据初步筛选

通过对原始数据附件 3.xlsx 分析，留言分为留言主题和留言详情，留言主题中清晰指出了特定地点以及特定人群，我们想要实现快速提取留言中的特定地点或人群，选择了对留言主题进行提取分析。

3.2.2jieba 分词

提取原始数据中的留言主题后，我们分别对提取到的这些留言进行分词，分词之后得到的数据如图 3.2 所示，这种分词结果直接用来进行文字向量化会出现问题，我们需要对这些分词结果做进一步处理得到如图 3.3 所示的结果。

```
[['A', '市', '经济', '学院', '体育', '学院', '变相', '强制', '实习'], ['在',
 'A', '市', '人才', 'app', '上', '申请', '购房', '补贴', '为什么', '通不过'],
  ['希望', '西地省', '把', '抗癌', '药品', '纳入', '医保', '范围'], ['A5', '
 区', '劳动', '东路', '魅力', '之', '城', '小区', '临街', '门面', '烧烤', '夜
 宵', '摊'], ['请', '给', 'K3', '县', '乡村', '医生', '发', '卫生室', '执业',
  '许可证'], ['A5', '区', '劳动', '东路', '魅力', '之', '城', '小区', '一楼',
  '的', '夜宵', '摊', '严重', '污染', '附近', '的', '空气'], ['A5', '区', '劳
动', '东路', '魅力', '之', '城', '小区', '一楼', '的', '夜宵', '摊', '严重',
  '污染', '附近', '的', '空气', ',', '急需处理', '!'], ['A', '市', '能否',
'设立', '南塘', '城轨', '公交站', '?'], ['请求', 'A', '市', '地铁', '2',
'#', '线', '在', '梅', '溪湖', 'CBD', '处', '增设', '一个', '站'], ['请问',
 'A', '市', '什么', '时候', '能', '普及', '5G', '网络', '?'], ['魅力', '之
', '城', '小区', '临街', '门面', '油烟', '直排', '扰民'], ['A5', '区', '劳动
', '东路', '魅力', '之', '城', '小区', '油烟', '扰民'], ['A', '市', '经济',
'学院', '寒假', '过年', '期间', '组织', '学生', '去', '工厂', '工作'], ['L',
```

图 3.2jieba 分词结果

```
['A 市 经济 学院 体育 学院 变相 强制 实习 ', '在 A 市 人才 app 上 申请 购房 补贴
 为什么 通不过 ', '希望 西地省 把 抗癌 药品 纳入 医保 范围 ', 'A5 区 劳动 东路 魅
 力 之 城 小区 临街 门面 烧烤 夜宵 摊 ', '请 给 K3 县 乡村 医生 发 卫生室 执业 许
 可证 ', 'A5 区 劳动 东路 魅力 之 城 小区 一楼 的 夜宵 摊 严重 污染 附近 的 空气
 ', 'A5 区 劳动 东路 魅力 之 城 小区 一楼 的 夜宵 摊 严重 污染 附近 的 空气 ， 急
 需处理 ！ ', 'A 市 能否 设立 南塘 城轨 公交站 ？ ', '请求 A 市 地铁 2 # 线 在
 梅 溪湖 CBD 处 增设 一个 站 ', '请问 A 市 什么 时候 能 普及 5G 网络 ？ ', '魅力
  之 城 小区 临街 门面 油烟 直排 扰民 ', 'A5 区 劳动 东路 魅力 之 城 小区 油烟 扰
 民 ', 'A 市 经济 学院 寒假 过年 期间 组织 学生 去 工厂 工作 ', 'L 市 物业 服务
 收费 标准 应 考虑 居民 的 经济 承受能力 ', 'A 市 江山 帝景 新房 有 严重 安全隐患
 ', 'A 市 魅力 之 城 小区 底层 商铺 营业 到 凌晨 ， 各种 噪音 好 痛苦 ', '12123
 上 申请 驾驶证 期满 换证 ， 一个 星期 了 都 无人 受理 ', 'A5 区 魅力 之 城 小区
```

图 3.3 进一步处理后的结果图

3.2.3 提取文本特征

词频－逆向文件频率（TF-IDF）是一种在文本挖掘中广泛使用的特征向量化方法，它可以体现一个文档中词语在语料库中的重要程度。词语由 t 表示，文档由 d 表示，语料库由 D 表示。词频 TF 是词语 t 在文档 d 中出现的次数。文件频率 DF 是包含词语的文档的个数。如果我们只使用词频来衡量重要性，很容易过度强调在文档中经常出现而并没有包含太多与文档有关的信息的词语。如果一个词语经常出现在语料库中，它意味着它并没有携带特定的文档的特殊信息。逆向文档频率数值化衡量词语提供多少信息。

我们这里利用的是 Python 中的第三方库 sklearn 来实现我们的 TF-IDF 算法，具体实现代码如图 3.4 所示。

```
# 该类会将文本中的词语转换为词频矩阵，矩阵元素a[i][j] 表示j词在i类文本下的词频
vectorizer = CountVectorizer(max_features=20000)
# 该类会统计每个词语的tf-idf权值
tf_idf_transformer = TfidfTransformer()
# 将文本转为词频矩阵并计算tf-idf
tfidf = tf_idf_transformer.fit_transform(vectorizer.fit_transform(message_list2))
# 获取词袋模型中的所有词语
tfidf_matrix = tfidf.toarray()
# 获取词袋模型中的所有词语
word = vectorizer.get_feature_names()
```

图 3.4 TF-IDF 具体实现代码

3.3 PCA 数据降维

降维就是一种对高维度特征数据预处理方法。降维是将高维度的数据保留下最重要的一些特征，去除噪声和不重要的特征，从而实现提升数据处理速度的目的。在实际的生产和应用中，降维在一定的信息损失范围内，可以为我们节省大量的时间和成本。降维也成为应用非常广泛的数据预处理方法[4]。

我们对数据进行向量化之后，在进行文本聚类之前要把数据降为二维数据，才能进行接下来的文本聚类操作，我们这里使用的是 Python 的机器学习库中自带的 PCA。

3.4 K-Means 文本聚类

K-means 基本算法比较简单。首先,选择 k 个初始中心，其中 k 是用户指定的参数，即所期望的簇的个数。每个点指派到最近的中心，而指派到一个中心的点集为一个簇。然后根据指派到簇的点，更新为每个簇的中心。重复指派和更新步骤，直到簇不发生变化，或等价地，直到中心不发生变化[5]。

前面介绍了 TD－IDF 我们可以通过用 TD－IDF 衡量每个单词在文件中的重要程度。如果多个文件，它们的文件中的各个单词的重要程度相似，我就可以说这些文件是相似的[6]。一种很自然的想法是用两者的欧几里得距离来作为相异度，欧几里得距离的定义如下：

$$d(X,Y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} \tag{16}$$

其意义就是两个元素在欧氏空间中的集合距离,因为其直观易懂且可解释性强，被广泛用于标识两个标量元素的相异度。我们可以将 X，Y 分别理解为两篇文本文件，xi，y 是每个文件单词的 TD－IDF 值。这样就可以算出两文件的相似度了。这样我们可以将文件聚类的问题转化为一般性的聚类过程，样本空间中的两点的距离可以欧式距离描述[6]。除欧氏距离外，常用作度量标量相异度的还有曼哈顿距离和闵可夫斯基距离，两者定义如下：

曼哈顿距离为：

$$d(X,Y)=|x_1-y_1|+|x_2-y_2|+\cdots+|x_n-y_n| \tag{17}$$

闵可夫斯基距离为：

$$d(X,Y)=\sqrt[p]{|x_1-y_1|^p+|x_2-y_2|^p+\cdots+|x_n-y_n|^p} \tag{18}$$

依次比较每一个对象到每一个聚类中心的距离，将对象分配到距离最近的聚类中心的类簇中，得到k个类簇$S_1,S_2,S_3,\ldots,S_k$ 。

K-means算法用中心定义了类簇的原型，类簇中心就是类簇内所有对象在各个维度的均值，其计算公式为

$$C_t=\frac{\sum_{X_i\in S_l}X_i}{|S_l|} \tag{19}$$

式中，C_l表示第个聚类的中心，$1\le l\le k$，S_l表示第l个类簇中对象的个数，X_i表

示第l个类簇中第i个对象，$1 \le i \le S_l$。算法流程如图3.5所示：

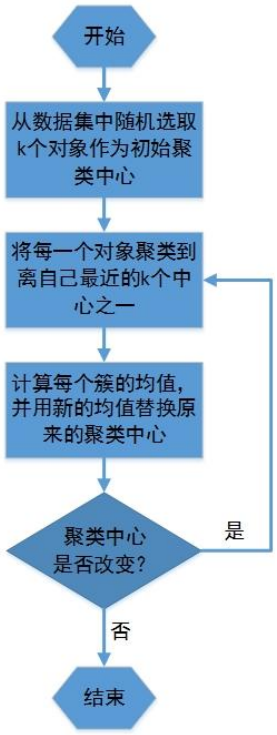

图3.5 K-means算法流程图

整个文本聚类过程可以先后分为两步：1、计算文本集合各个文档中 TD-IDF 值；2、根据计算的结果，对文件集合用 K-means 聚类方法进行迭代聚类[6]。

我们对提取到的文本进行 K-means 聚类，得到的聚类结果清晰显示的每一类的距离，如图 3.6 所示，但是这样的聚类结果也有缺点，从图中可以看出各种颜色之间的距离非常小，这样分出来的这一类就会非常接近，但还是有很多非常离散的点，这几类的文本并没有非常相似，因此我们还要对聚类之后的文本做进一步的筛选分离。并且数据量非常庞大，从图中只能看出大概的聚类成效，很多重合的点我们肉眼看不出来，由此我们可以初步观察留言的分布情况（具体实现程序见附件程序：test2.py）。

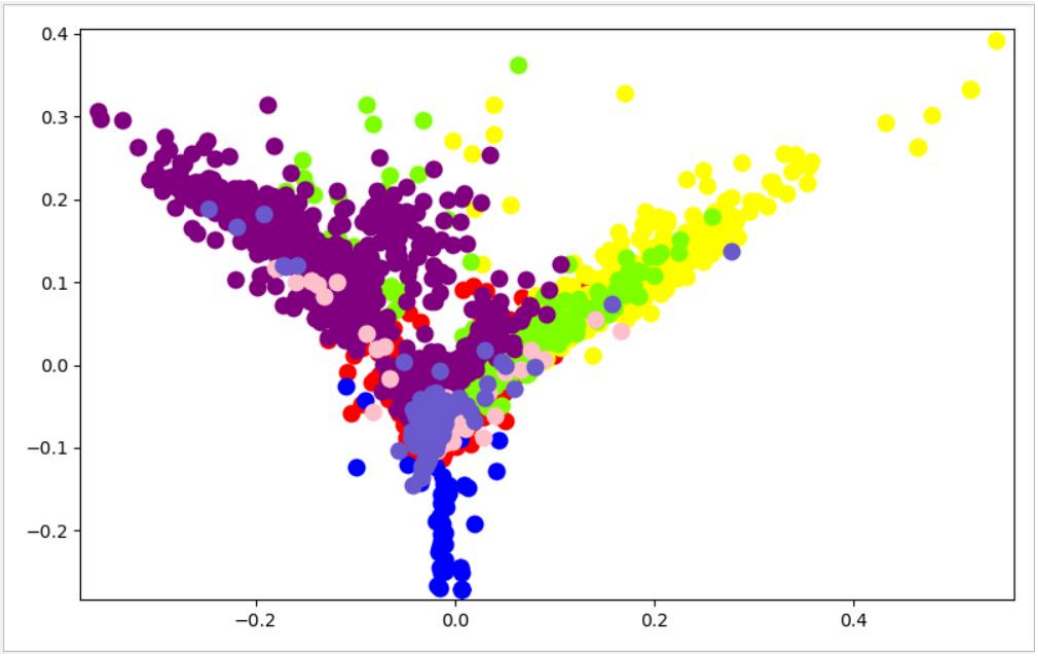

图 3.6 聚类结果

3.5 设置热点评价标准

对留言进行文本聚类之后，我们设置了相应的热度评价标准，对某一特定地点或人群中产生的问题进行数量统计，并对该类的问题的点赞数量和反对数量做求和统计，我们选用提出问题的人数和交互数量总和为热度评价指数的评价标准，并对这两个影响因素赋予各自的权重进行计算，由于避免交互数量过大引起的热点指数过大，我们将交互数量用 ln 函数做平滑处理，减少误差，具体公式如下所示。

热度指数=提出问题人数*0.8+ln（交互数量） *0.2

交互数=点赞数+反对数

3.6 问题 2 结果分析

3.6.1 生成表 2 热点问题留言明细表

我们对文本聚类之后得到结果每一类文本加上相同的 ID，生成热点问题留言明细表，部分表格如表 3.1 所示（完整表格见附件：热点问题留言明细表.xlsx），可以清晰的显示哪些问题属于同一类，并可以对这些问题进行逐一了解逐一排查

（具体实现程序见附件程序：java 文件夹）。

表 3.1 热点问题留言明细表

问题 ID	留言编号	留言用户	点赞数	反对数
1	188006	A000102948	0	0
2	188007	A00074795	1	0
2	198975	A00051906	4	9
3	188031	A00040066	1	0
3	192685	A909099	1	0
4	188039	A00081379	1	0
5	188059	A00028571	0	0
5	191327	A00073692	0	0
5	209549	A0007057	0	0
5	215834	A00028571	0	0
5	219977	A00088233	0	0
5	223788	A00024012	0	0
5	234800	A00046970	0	0
5	256379	A00086602	0	0
5	265914	A00020501	0	0
5	273942	A000106463	0	0
6	188073	A909164	0	0
7	188074	A909092	0	0
8	188119	A00035029	0	0

从表格中我们可以清晰的得出，这些群众反映激烈的问题处于哪些方面，哪些地点。

3.6.2 生成表 1 热点问题表

我们对已经生成的热点问题留言明细表进行处理，相同 ID 的问题留言提取主要人物及其主要问题，并根据已经制定好的热度评价指标计算热度指数，将得到的热度指数进行降序排序，热度指数最高则就为当前热度最高的问题，数据如表 3.2 所示(具体实现程序见附件程序：sort.py)。

表 3.2 热点问题表

热度排名	问题ID	热度指数	时间范围	地点/人群	问题描述
1	115	24.83	2019/02/10 至 2019/12/16	A 市公交线路	关于给 A 市公交线路的建议
2	86	23.64	2019/07/18 至 2019/08/07	A 市伊景园滨河苑	A 市伊景园滨河苑项目捆绑销售车位
3	34	22.36	2019/11/13 至 2019/12/26	A 市 A2 区丽发新城小区	A 市丽发新城小区附近搅拌站噪音扰民和污染环境
4	117	19.73	2019/01/12 至 2019/12/15	A7 县泉塘小区	A7 县泉塘小区非法住改商，开设麻将馆扰民
5	358	18.48	2019/01/10 至 2019/09/03	A 市在职人员	咨询 A 市相关部门各类政策改革

从表格中我们可以得出，当前社会中存在的主要问题的主要分布方向及其产生问题的主要人群，以及当前问题的热度和发生时间。

4 问题 3 分析方法与过程

对于附件 4 中相关部门对留言的答复意见，我们分别从答复的相关性、完整性、可解释性等角度分析了该数据，并对答复意见作出了质量评价。流程如图 4.1 所示。

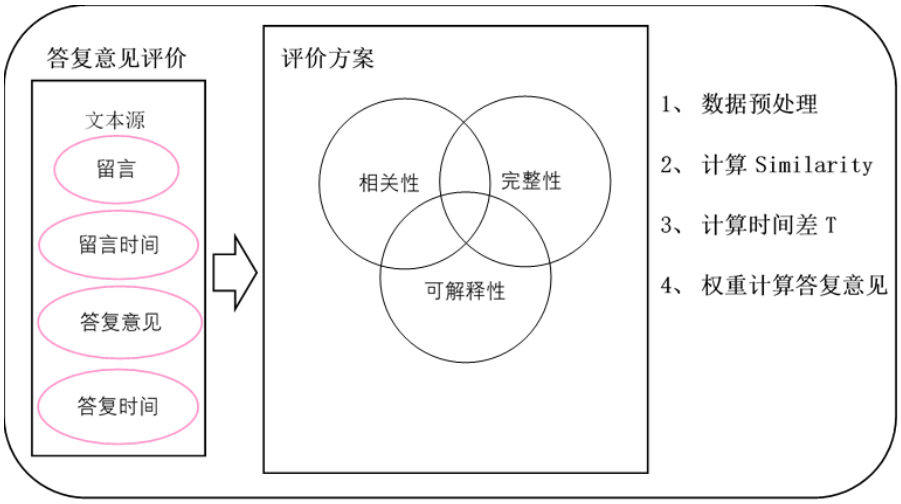

图 4.1 问题 3 流程图

4.1 向量空间余弦相似度

相似度度量（Similarity），即计算个体间的相似程度，相似度度量的值越小，说明个体间相似度越小，相似度的值越大说明个体差异越大[7]。

对于多个不同的文本或者短文本对话消息要来计算他们之间的相似度如何，一个好的做法就是将这些文本中词语，映射到向量空间，形成文本中文字和向量数据的映射关系，通过计算几个或者多个不同的向量的差异的大小，来计算文本的相似度[7]。

我们这里使用的是一种成熟的向量空间余弦相似度方法(Cosine Similarity)计算相似度。余弦相似度用向量空间中两个向量夹角的余弦值作为衡量两个个体间差异的大小。余弦值越接近 1，就表明夹角越接近 0 度，也就是两个向量越相似，这就叫“余弦相似性”[7]。其计算公式为：

$$similarity = \cos(\theta) = \frac{\sum_{i=1}^{n}(x_i * y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2} * \sqrt{\sum_{i=1}^{n}(y_i)^2}} \qquad (20)$$

思路上就是：将文本中的词汇映射到向量空间，来计算两个向量的夹角余弦值，作为两个文本相似度的判断[7]。余弦相似性是指通过测量两个向量内积空间的夹角的余弦值来度量它们之间的相似性。0 度角的余弦值是 1，而其他任何角度的余弦值都不大于 1;并且其最小值是-1。从而两个向量之间的角度的余弦值确定两个向量是否大致指向相同的方向。两个向量有相同的指向时，余弦相似度的值为 1；两个向量夹角为 90° 时，余弦相似度的值为 0；两个向量指向完全相

反的方向时，余弦相似度的值为-1。在比较过程中，向量的规模大小不予考虑，仅仅考虑到向量的指向方向。余弦相似度通常用于两个向量的夹角小于 90° 之内，因此余弦相似度的值为 0 到 1 之间。

产生的相似性范围从-1 到 1:-1 意味着两个向量指向的方向正好截然相反，1 表示它们的指向是完全相同的，0 通常表示它们之间是独立的，而在这之间的值则表示中度的相似性或相异性。

4.2 数据预处理

我们在提取了附件 4 中的留言及留言答复之后，需要对提取的两个字符串做相应的处理才能进行后续的相似度度量操作。

我们在处理字符串的时候要判断字符串中的字符是否为汉字，对于字符串中的数字和标点我们不做处理，并获取该字符对应的 GB2312 编码，将两个字符中的中文字符以及出现的总数封装到 AlgorithmMap 中。获取某个字符的 GB2312 编码的处理过程中，我们的判断方法为：正常情况下 buffer 应该是两个字节，否则说明改字符不属于 GB2312 编码，否则则返回-1，说明不认识当前字符具体实现代码如图 4.1 所示。

```
/**
 *根据输入的Unicode字符，获取它的GB2312编码或者ascii编码，
 *
 * @param ch 输入的GB2312中文字符或者ASCII字符(128个)
 * @return ch在GB2312中的位置，-1表示该字符不认识
 */
public static short getGB2312Id(char ch) {
    try {
        byte[] buffer = Character.toString(ch).getBytes("GB2312");
        if (buffer.length != 2) {
            // 正常情况下buffer应该是两个字节，否则说明ch不属于GB2312编码，故返回-1，此时说明不认识该字符
            return -1;
        }
        int b0 = (int) (buffer[0] & 0x0FF) - 161; // 编码从A1开始，因此减去0xA1=161
        int b1 = (int) (buffer[1] & 0x0FF) - 161;
        return (short) (b0 * 94 + b1);// 第一个字符和最后一个字符没有汉字，因此每个区只收16*6-2=94个汉字
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    return -1;
}
```

图 4.1 获取该字符对应的 GB2312 编码代码

在获取字符对应的 GB2312 编码代码，我们要讲两个字符串中的中文字符以及出现的总数封装到 AlgorithmMap 中，具体实现代码如图 4.2 所示。

```java
public static double getSimilarity(String doc1, String doc2) {
    if (doc1 != null && doc1.trim().length() > 0 && doc2 != null&& doc2.trim().length() > 0) {
        Map<Integer, int[]> AlgorithmMap = new HashMap<Integer, int[]>();
        //将两个字符串中的中文字符以及出现的总数封装到，AlgorithmMap中
        for (int i = 0; i < doc1.length(); i++) {
            char d1 = doc1.charAt(i);
            if(isHanZi(d1)){//标点和数字不处理
                int charIndex = getGB2312Id(d1);//保存字符对应的GB2312编码
                if(charIndex != -1){
                    int[] fq = AlgorithmMap.get(charIndex);
                    if(fq != null && fq.length == 2){
                        fq[0]++;//已有该字符，加1
                    }else {
                        fq = new int[2];
                        fq[0] = 1;
                        fq[1] = 0;
                        AlgorithmMap.put(charIndex, fq);//新增字符入map
                    }
                }
            }
        }
        for (int i = 0; i < doc2.length(); i++) {
            char d2 = doc2.charAt(i);
            if(isHanZi(d2)){
                int charIndex = getGB2312Id(d2);
                if(charIndex != -1){
                    int[] fq = AlgorithmMap.get(charIndex);
                    if(fq != null && fq.length == 2){
                        fq[1]++;
                    }else {
                        fq = new int[2];
                        fq[0] = 0;
                        fq[1] = 1;
                        AlgorithmMap.put(charIndex, fq);
                    }
                }
            }
        }
```

图 4.2 封装 AlgorithmMap 具体代码

4.3 Cosine Similarity 具体实现

在对获取到的数据进行初步的数据预处理时候，我们就可以对获取的两个字符串进行 Cosine Similarity 的计算，我们将文本映射到向量空间，形成中文文本和向量之间的映射关系，从而计算出几个或者多个不同的向量的差异的大小，得到相应的 Cosine Similarity 值，具体的实现代码如图 4.3 所示。

```java
Iterator<Integer> iterator = AlgorithmMap.keySet().iterator();
double sqdoc1 = 0;
double sqdoc2 = 0;
double denominator = 0;
while(iterator.hasNext()){
    int[] c = AlgorithmMap.get(iterator.next());
    denominator += c[0]*c[1];
    sqdoc1 += c[0]*c[0];
    sqdoc2 += c[1]*c[1];
}
return denominator / Math.sqrt(sqdoc1*sqdoc2);//余弦计算
```

图 4.3 Cosine Similarity 具体实现代码

4.4 设立答复意见质量评价标准

通过对答复意见 Cosine Similarity 值的计算，我们将对总体答复意见质量通过如下公式进行计算：

$$Score = 0.7 * \text{Cosine Similarity} + 0.3 * \frac{1}{T} \quad (21)$$

其中，

$$T = \text{答复时间} - \text{留言时间（单位：天）}$$

通过整体的计算我们得到附件 4 的整体答复意见质量指数。

4.5 问题 3 结果分析

我们通过计算生成了答复意见质量指数表，指数越高表示质量越高，表格部分数据如表 4.1 所示（完整表格见附件过程数据：问题 3 数据.xlsx；具体实现程序见附件程序：java 文件夹）。

表 4.1 部分答复意见指数

留言编号	留言用户	Similarity	时间差	答复意见质量指数
2549	A00045581	0.605251204	15.22550926	0.443379617
2554	A00023583	0.247196654	14.73993056	0.193390535
2555	A00031618	0.593569453	14.75636574	0.435828826
2557	A000110735	0.426521619	14.77930556	0.318863787
2574	A0009233	0.527058081	15.70012731	0.388048782
2759	A00077538	0.234650621	31.05888889	0.173914505
2849	A000100804	0.516945333	40.93443287	0.369190526
3681	UU00812	0.457462448	28.52153935	0.33074208
3683	UU008792	0.373343841	16.23244213	0.279822196
3684	UU008687	0.566630598	16.23965278	0.41511472
3685	UU0082204	0.468160248	70.73336806	0.331953453
3692	UU008829	0.53755447	30.47511574	0.386132226

从表格中我们可以清晰的看出每一条留言的答复意见质量指数，为了能够更加清晰的显示答复意见的总体质量，我们针对答复意见质量指数抽取其中 200 条生成了答复意见质量指数折线图，折线图如图 4.4 所示。

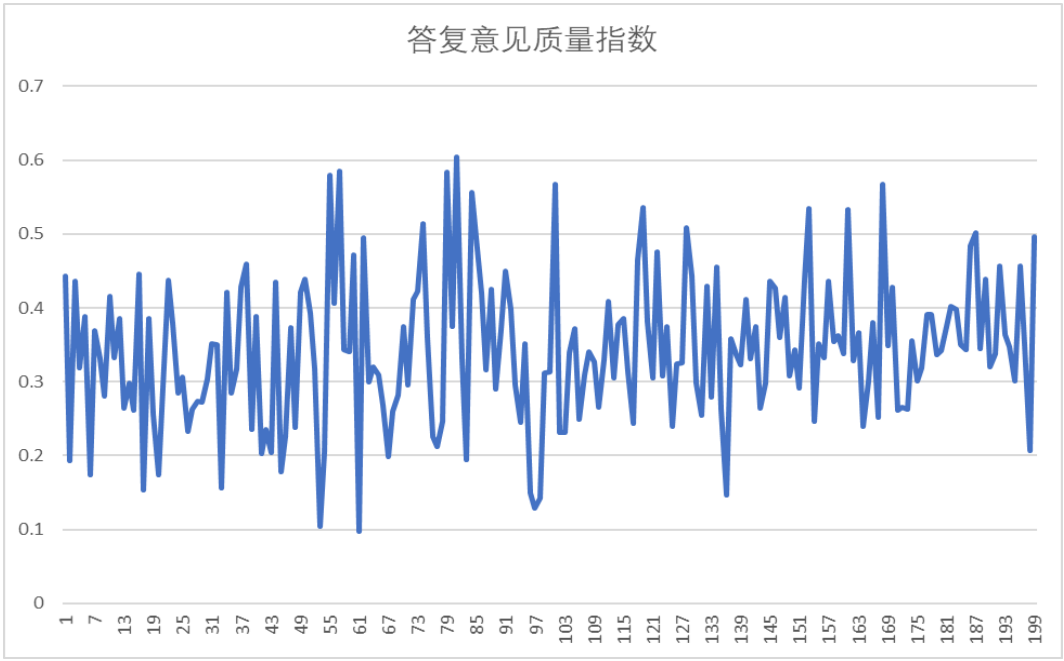

图 4.4 答复意见质量指数折线图

从图中我们可以看出，针对附件 4 中的留言和答复意见，极少数答复意见质量指数在 0.5-0.6 左右，极大部分答复意见质量指数集中在 0.2-0.5 之间。

5 总结与展望

首先我们对所提供的数据进行分析，通过运用朴素贝叶斯分类器对留言进行分类，以及运用 K-Means 聚类方法将热点问题进行归类，再将全部数据分为训练集和测试集来测试我们的分类模型以及对分类模型进行评价，可以得到训练集的整体准确率达到 99.6%，测试集的整体准确率达到 99.7%。然后，我们先对原始数据进行初步筛选，通过 TF-IDF 算法提取文本关键字，之后运用 PCA 降维将数据降为二维数据，再利用 K-Means 文本聚类方法把相同问题的留言归为一类，每一类求热点指数，排序求前五名。最后，通过对答复意见的 Cosine Similarity 值和时间差的计算得到答复意见质量指数，生成了答复意见质量指数表。但由于数据量过大，程序运行时间过长，在将来的工作中我们将不断完善模型提高整体准确率以及缩短运行时间。

6 参考文献

[1] 钟熙,孙祥娥.基于 Kmeans++聚类的朴素贝叶斯集成方法研究[J].计算机科学,2019,46(S1):439-441+451.

[2] HAN J W,KAMBER M.数据挖掘概念与技术[M].范明，孟小锋，译.北京：机械工业出版社，2000:173-175.

[3] 许甜华,吴明礼.一种基于 TF-IDF 的朴素贝叶斯算法改进[J].计算机技术与发展,2020,30(02):75-79.

[4] Microstrong0305. 主成分分析（PCA）原理详解.CSDN.(2018-06-09).https://blog.csdn.net/program_developer/article/details/80632779

[5] 姚明宇. 基于 k-means 的中文文本聚类算法[C]. Civil Aviation University of China.Proceedings of 2010 International Conference on Services Science, Management and Engineering(Volume 2).Civil Aviation University of China:智能信息技术应用学会,2010:20-23.

[6] yyphlj1. 基于 K-means 算法的文本聚类.CSDN.(2015-07-08).https://blog.csdn.net/yyphlj1/article/details/46805061

[7] chengwangbaiko.Java 实现余弦定理计算文本相似度.CSDN.(2017-12-07).https://blog.csdn.net/chengwangbaiko/article/details/78742237