基于迁移学习和 TF-IDF 的文本挖掘应用

摘　要

大数据、云计算、人工智能等技术发展日新月异，建立基于自然语言处理技术的智慧政务系统对提升政府的管理水平和施政效率具有极大的推动作用。本文基于 BERT 迁移学习及密度聚类的思想，结合双向 GRU 网络、TF-IDF 等解决了文本分类和热点挖掘的等问题，并利用余弦相似度从相关性的角度对答复意见的质量给出了一套评价方案。

对于问题一，在数据预处理阶段，本文先将群众留言按字进行分词和去停用词等。在此基础上，利用 BERT 的迁移模型对预处理后的群众留言进行句子层面的特征表示，最后将获得的特征向量放入双向 GRU 网络中再接入 Softmax 分类器进行分类。和其他卷积神经网络对比，双向 GRU 网络有效防止了在小数据上面的过拟合问题，结果表明分类准确率高达 90%左右。

对于问题二，在假设某一时段内群众集中反映的某一问题其主题和使用的关键字大致相同的前提下，本文首先对留言详情和主题分别进行分词和去停用词等预处理，得到了更为重要的关键字。然后使用 TF-IDF 算法把留言转化相应的向量，最后利用密度聚类有效地将热点问题筛选出来。

对于问题三，考虑到应付性的回复以及与无效回复普遍出现，本文选取留言与回复内容的相关性作为回复的质量评估指标。首先对于留言和回复内容去除所有字母、数字、标点、并去除停用词和客套词并进行分词处理。然后根据两个文本提取出来的词生成的两个向量计算其余弦相似度值，并将其作为相关度的评判依据生成了一套评价方案。

关键词：BERT 迁移学习；双向 GRU 网络；密度聚类；TF-IDF 算法；余弦相似度

Abstract

Big data, cloud computing, artificial intelligence and other technologies are developing rapidly. The establishment of an intelligent government affairs system based on natural language processing technology will greatly promote the government's management level and efficiency. Based on BERT's migration learning and density clustering, this paper solves the problems of text classification and hot spot mining by combining bidirectional GRU network, TF-IDF, etc. And gives a set of evaluation scheme for the quality of replies from the perspective of correlation by using cosine similarity.

As for question 1, in the data pre-processing stage, this paper first divides the message of the masses into words by words and stops words. On this basis, BERT's migration model was used to represent the characteristics of the pre-processed crowd comments at the sentence level. Finally, the acquired feature vectors were put into the bidirectional GRU network and then connected to the Softmax classifier for classification. Compared with other convolutional neural networks, the bidirectional GRU network effectively prevents the problem of overfitting on small data, and the results show that the classification accuracy is up to about 90%.

For question 2, assuming that the topic and the keywords used for a certain problem reflected by the masses in a certain period of time are roughly the same, this paper firstly preprocessed the message details and the topic by word segmentation and stop and stop words respectively, and obtained more important keywords. Then the TF-IDF algorithm is used to transform the message into the corresponding vector, and finally the hot issues are effectively screened out by density clustering.

For question 3, considering that the response is coping and the common occurrence of invalid reply, this paper selects the correlation between the message and the reply content as the quality evaluation index of the reply. First, remove all letters, Numbers, punctuation, and stop words and polite words for the message and reply content and word segmentation. Then, based on the two vectors generated by the two words extracted from the text, the cosine similarity value is calculated, and a set of evaluation scheme is generated by taking it as the evaluation basis of relevance.

KeyWords: BERT transfer learning; Bidirectional GRU network; Density clustering; TF - IDF algorithm; Cosine similarity

目录

一、问题的提出

1.1 问题的背景

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。

1.2 已知的条件

附件给出了收集自互联网公开来源的群众问政留言记录，及相关部门对部分群众留言的答复意见。

1.3 问题的提出

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。请利用自然语言处理和文本挖掘的方法解决下面的问题：

问题一：群众留言分类，即根据附件 2 给出的数据，建立关于留言内容的一级标签分类模型。

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”。

问题三：答复意见评价，针对附件 4 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一套评价方案，并尝试

实现。

二、问题的分析

针对问题一，为了方便后续的处理，可以考虑先将群众留言进行分词和去停用词等预处理。然后再通过词向量的方式进行文本表示，即可把预处理后的群众留言转换成计算机可理解的方式。基于 BERT 模型的基础上，结合 GRU 算法可以将群众留言进行分类并有效防止在小数据上面的过拟合问题。

针对问题二，可以假设某一时段内群众集中反映的某一问题，其主题和使用的关键字大致相同，因此，我们对留言详情和留言主题进行分词和去停用词等预处理，得到更为重要的关键字，从而使用 TF-IDF，把不同的留言转化为不同的向量，而某一类问题所映射的向量更倾向聚在一起，从而我们使用密度聚类（DBSCAN）可以有效的将热点问题筛选出来。

针对问题三，考虑到有很多应付性的回复以及与留言内容毫不相干的回复，我们可以考虑留言与回复内容的相关性对回复的质量进行评估。对于留言和回复内容我们去除所有字母、数字、标点、并去除停用词和客套词，然后进行分词处理，并根据两个文本提取出来的词生成两个向量，计算其余弦值作为相关度的评判依据。这样可以把应付性以及毫无作用的回复筛选出来，不过也会造成一些可靠回复的误判。

三、符号说明

符号	符号说明
i	某一个样本
j	某一个类别标签
k	类别数
m	样本个数
n	维度
p	条件概率
x	测试输入

A	向量
E	字的文本输入
$\|D\|$	文档总数
D_j	j 类别的文档
T	某一词语
x^i	第 i 个训练样本对应的短文本向量
$N_{i,j}$	词语 T_i 在文档 D_j 中出现的次数
θ	各个类别对应的分类参数

四、模型的建立与求解

4.1 文本预处理

为了能把群众留言放到后面的模型训练，把相关数据导入之后将一级分类标签转化成相应的数字。

	留言编号	留言用户	留言主题	留言时间	留言详情	一级标签	y
0	24	A00074011	A市西湖建筑集团占道施工有安全隐患	2020/1/6 12:09:38	\n\t\t\t\t\t\n\t\t\t\t\tA3区大道西行便道，未管所路口至加油站路段，...	城乡建设	0
1	37	U0008473	A市在水一方大厦人为烂尾多年，安全隐患严重	2020/1/4 11:17:46	\n\t\t\t\t\t\n\t\t\t\t\t位于书院路主干道的在水一方大厦一楼至四楼人为...	城乡建设	0
2	83	A00063999	投诉A市A1区苑物业违规收停车费	2019/12/30 17:06:14	\n\t\t\t\t\t\n\t\t\t\t\t尊敬的领导：A1区苑小区位于A1区火炬路，小...	城乡建设	0
3	303	U0007137	A1区蔡锷南路A2区华庭楼顶水箱长年不洗	2019/12/6 14:40:14	\n\t\t\t\t\t\n\t\t\t\t\tA1区A2区华庭小区高层为二次供水，楼顶水箱...	城乡建设	0

图 1 数据预处理示意图

将 train、test、val 按照 0.7:0.24:0.06 的比例划分为训练集和验证集，应用深度学习模型需要将文本进行截断补齐，留言主题最长的字段只有 48 个字，因此取这里的截断补齐长度为 50。处理文本数据前需要对文本数据进行分词、去停用词、关键词提取等处理[1]。考虑到群众留言的字长小，本文采取按字分类的方式分词。停用词是指其他词相比那些功能极其普遍，没有什么实际含义的词，它们通常是一些单字，单字母以及高频的单词，比如中文中的“是、的、地、了、

吗”等，英文中的“a、is、this、the、of”等[2]。对于停用词进行处理，可以降低噪音，节省储存空间可计算时间[3]。

4.2 群众留言分类问题模型的建立与求解

考虑到群众留言这类的短文本由于特征稀疏，不能用字符或者词语表示出完整的语义，导致其特征表示向量不能够很好的代表短文本语义，而本文中而基于 BERT 模型的群众留言分类能解决这个问题。其核心思想为使用预训练语言模型对短文本进行句子层面上的特征表示，由于 BERT 预训练语言模型具有强大的表义能力，本文使用该模型对预处理后的短文本进行句子层面的特征表示，最后将获得的特征向量放入双向 GRU 网络中再接入 Softmax 分类器进行分类。

4.2.1 BERT 模型的建立

BERT（Bidirectional EncoderRepresentations from Transformers），其重要部分是基于双向 Transformer 编码器实现的，其模型结构见图 2。

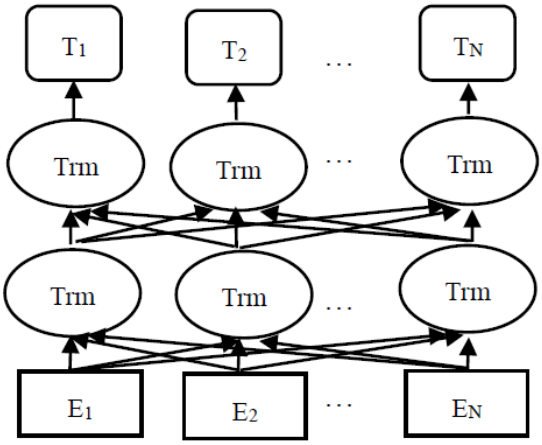

图 2 BERT 模型结构示意图

其中，E_1，$E_2 \cdots E_N$ 表示字的文本输入，通过双向的 Transformer 编码器得到文本的向量化表示。和大多数 seq2seq 模型一样，transformer 的结构也是 encoder 和 decoder 组成，即输入是一个序列，输出也是一个序。其中，Encoder 将一个可变长度的输入序列变为固定长度的向量，Decoder 将这个固定长度的向量解码成可变长度的输出序列，模型结构见图 3。

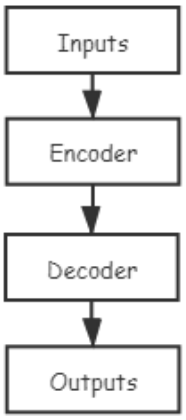

图 3 Transformer 模型结构示意图

4.2.2 双向 GRU 神经网络

Cho 在 LSTM 的基础上提出了 GRU 模型[4]，对 LSTM 模型进行了很多简化，LSTM 中的遗忘门和输入们用更新门来替代,GRU 单元结构见图 4。

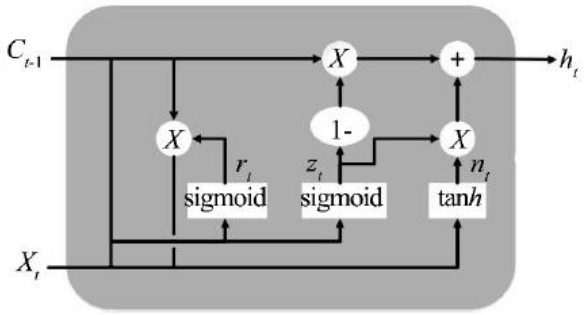

图 4 GRU 单元结构示意图

GRU 单元的前向计算公式如下

$$z_t = sigmoid\left(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}\right)$$
$$r_t = sigmoid\left(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}\right)$$

$$n_t = \tanh\left(W_{in}x_t + b_{in} + r_t \times \left(W_{hn}h_{t-1} + b_{hn}\right)\right)$$

$$h_t = \left(1 - z_t\right) \times n_t + z_t \times h_{t-1}$$

考虑到信息提取情况下只有前边输入还不够，双向 GRU (Bi-directional GRU) 各个单元都有不同的参数，正向计算和反向计算不共享权重矩阵,见图 5。通过对长期信息和短期信息的记忆，可以充分利用上下文信息，实现自动序列标注

和信息提取。

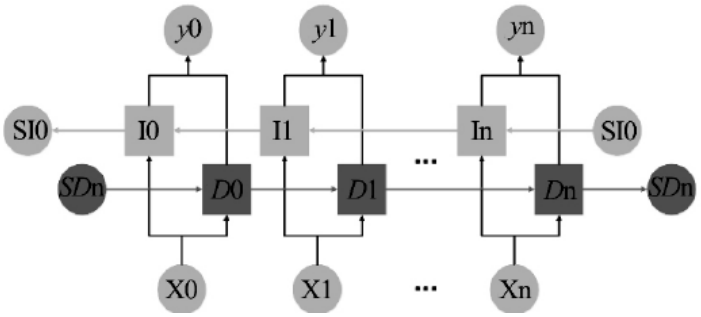

图 5 双向 GRU 结构示意图

4.2.3 Softmax 回归模型

Softmax 回归模型是一种广义的线性模型，是Logistic 回归模型在多分类问题上的推广，可以将获得的特征向量接入Softmax分类器进行分类表示[5]。假设有训练样本集$\left\{\left(x^1,y^1\right),\left(x^2,y^2\right),\ldots,\left(x^m,y^m\right)\right\}$。其中，$x^i\in R^n$为第$i$个训练样本对应的短文本向量，维度为$n$，共$m$个训练样本。而$y^i\in\{1,2,\ldots,k\}$为第$i$个训练样本对应的类别,$k$为类别个数。考虑到本文研究的是群众留言的问题并分成了七大类，因此类别数$k=7$。那么，对于给定的测试输入x，其分布函数为条件概率$p\left(y=j\mid x\right)$，且x出现概率最大的那一类别就是其所属的类别。在此基础上，再对其做归一化处理。

那么，Softmax 回归模型的判别函数$h_\theta\left(x\right)$为

$$h_\theta\left(x^i\right)=\begin{bmatrix}p\left(y^i=1\mid x^i;\theta\right)\\p\left(y^i=2\mid x^i;\theta\right)\\\vdots\\p\left(y^i=k\mid x^i;\theta\right)\end{bmatrix}=\frac{1}{\sum_{j-1}^{k}e^{\theta_j^T x^i}}\begin{bmatrix}e^{\theta_1^T x^i}\\e^{\theta_2^T x^i}\\\vdots\\e^{\theta_k^T x^i}\end{bmatrix}$$

其中，$\theta_1,\theta_2,\ldots,\theta_k\in R^n$ 为各个类别对应的分类参数。

$$\theta=\left[\theta_1^T,\theta_2^T,\ldots,\theta_k^T\right]^T$$

上述矩阵的每一行是一个类别对应的分类器参数。

用极大似然法来求解 Softmax 回归模型时，相应的似然函数为

$$L(\theta)=\prod_{i=1}^{m}\prod_{j=1}^{k}\left(\frac{e^{\theta_j^T x^i}}{\sum_{j=1}^{k} e^{\theta_j^T x^i}}\right)^{I\{y^i-j\}}$$

其中

$$I\{y^i=j\}=\begin{cases}1, y^i=j\\0, y^i\neq j\end{cases}$$

对数似然函数为

$$I(\theta)=\ln\left(L(\theta)\right)=\sum_{i=1}^{m}\sum_{j=1}^{k} I\{y^i=j\}\cdot\ln\frac{e^{\theta_j^T x^i}}{\sum_{j-1}^{k} e^{\theta_j^T x^i}}$$

Softmax 回归模型利用最小化损失函数求得θ的数值来预测一个新样本的类别，下面定义其损失函数为

$$J(\theta)=-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} I\{y^i=j\}\ln\frac{e^{\theta_j^T x^i}}{\sum_{l=1}^{k} e^{\theta_l^T x^i}}\right]$$

其中k表示类别标签的个数，m为样本个数，i表示某一个样本，j表示某一个类别标签，x^i表示第i个样本的向量表示。

进一步使用随机梯度下降法优化上述损失函数，考虑到样本x属于类别j的概率为

$$p\left(y^i=j\mid x^i;\theta\right)=\frac{e^{\theta_j^T x^i}}{\sum_{l=1}^{k} e^{\theta_{l=1}^T x^i}}$$

那么，定义损失函数的梯度为：

$$\nabla_{\theta_j} J(\theta)=-\frac{1}{m}\left(\sum_{i=1}^{m}\left[x^i\left(I\{y^i=J\}-p\left(y^i=j\mid x^i;\theta\right)\right)\right]\right)$$

每一次参数按$\theta_j=\theta_j-\alpha\nabla_{\theta_j}J(\theta),(j=1,2,\ldots,k)$进行更新迭代，最后即可求出$\theta$的具体数值得到判别函数$h_\theta(x)$进而实现对新的输入数据进行预测分类。

5.2.4 群众留言问题的求解

将预处理后的短文本放入 BERT 模型中对其进行句子层面的特征表示，得到了相应的二维向量，再将获得的特征向量放入神经网络中进行微调，具体流程见图 6。为了得到更好的网络结构，本文分别尝试了 LSTM、双向 LSTM、GRU、双向 GRU 和 TCNN 五种构建深度模型的方法，通过不同模型都训练 100 个 epoch 后，得到训练集和测试集上的准确率结果见图 7。

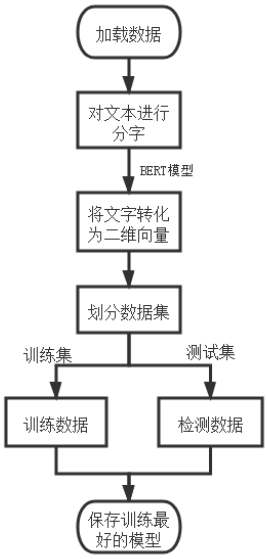

图 6 BERT 算法流程图

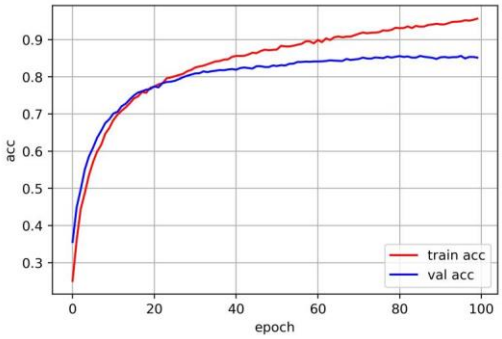

图 7 LSTM 在训练集和测试集上面的准确率

实验证明了双向 GRU 的效果较好，归结原因如下，一是 GRU 需要训练的参数更少；二是模型中加入 dropout 后，使得模型不易过拟合。数据对比见表 1。

表 1 各种神经网络方法效果对比表

	双向 LSTM	双向 GRU	TCNN	LSTM	GRU
train	0.977	0.97	0.9981	0.9752	0.9715
test	0.8595	0.8734	0.8371	0.8199	0.8538
F1-score	0.8559	0.8739	0.8375	0.8209	0.8528

同时，利用 F1 值最高的双向 GRU 模型做了相应的混淆矩阵，见图 8。

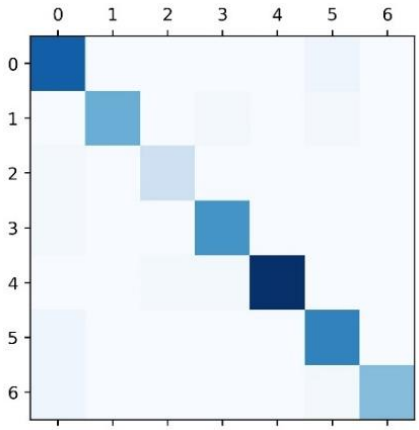

图 8 双向 GRU 在 val 上得到的混淆矩阵

4.3 热点问题挖掘

4.3.1 数据预处理

读取数据后先进行数据清洗，剔除了四条 2019 年之前数据（留言编号：188006、188007、188031、188039）见图 9。

	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
4159	286572	A23525	请求A市地铁2#线在梅溪湖CBD处增设一个站	2018-10-27 15:13:26	\n\t\t\t\t\t\n\t\t\t\t\t领导好！今天游了A3区山，在A3区山下湖大内...	3	0
4269	289408	A0012413	在A市人才app上申请购房补贴为什么通不过	2018-11-15 16:07:12	\n\t\t\t\t\t\t\t我叫朱琦梦，是2017年12月落户，并于2018年初在A...	0	0
4324	360113	A3352352	A市经济学院强制学生外出实习	2018-05-17 08:32:04	\n\t\t\t\t\t\t\tA市经济学院强制16届电子商务跟企业物流专业实习。其中我...	3	0
4325	360114	A0182491	A市经济学院体育学院变相强制实习	2017-06-08 17:31:20	\n\t\t\t\t\n\t\t\t\t\t　　书记您好，我是来自西地省经济学院体育学院...	9	0
0	188006	A000102948	A3区一米阳光婚纱艺术摄影是否合法纳税了？	2019/2/28 11:25:05	\n\t\t\t\t\n\t\t\t\t\t座落在A市A3区联丰路米兰春天G2栋320，...	0	0

图 9 异常数据示意图

在此基础上，再使用 Python 开发的中文分词模块 jieba 库对文本进行分词（精确模式或全模式）。由于中文文本词与词之间没有明显的界限，分词处理后便于提取词语。再对分词后的结果去停用词，与常规的去停用词略有不同。在停用词中，我们加入了一些诸如“市”,“区”,“山”,“路”,“请”,“恳请”,“希望”,“郡”,“县”,“街”等，但是对于数字和字母的组合我们并没有删除，部分结果见图 10。

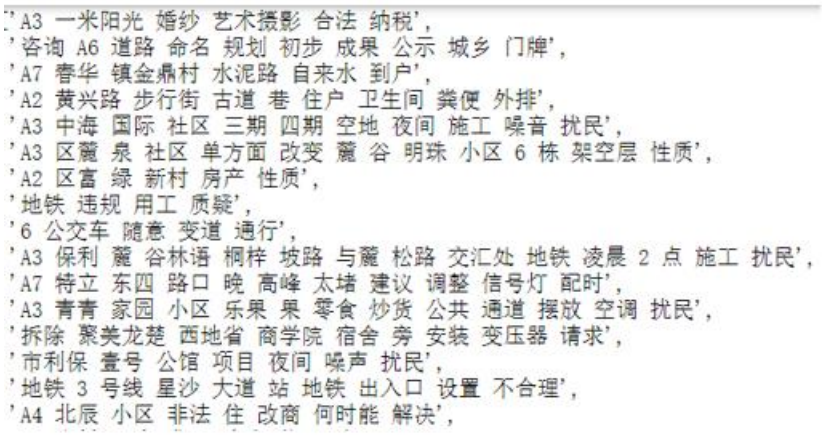

图 10 数据处理后部分结果示意图

4.3.2 TF-IDF 算法原理

在信息检索的过程中，TF-IDF（词频-逆文档频率）是一种用来评价单次在语料库或文档集合中重要程度的统计方法[6]。TF 和 IDF 的值的大小分别和词单次在文档集合和语料库中出现的次数呈正相关关系，这种方法的各个改进版本是搜索引擎给定用户查询时对文档的相关性进行排序和评分的重要工具。

TF（Term Frequency）表示词在给定文档中出现的频率，其值和词语的重要性呈正相关关系。文档 D_j 中的词语 T_i 其词频为

$$TF_{i,j} \equiv \frac{N_{i,j}}{\sum_k N_{k,j}}$$

其中，$N_{i,j}$ 表示词语 T_i 在文档 D_j 中出现的次数，$\sum_k N_{k,j}$ 表示文档 D_j 中所有词语的出现次数总和。

IDF（Inverse Document Frequency）其值越大，表明词语在整个文档集层面上具有很好的类别区分能力，它能用于评价对于整个文档集而言该词语重要程度。词语 T_i 其 IDF 为

$$IDF_i \equiv \frac{|D|}{\left|\{j : t_i \in d_j\}\right|}$$

其中，$|D|$ 表示语料库中所有文档总数，$\left|\{j : t_i \in d_j\}\right|$ 表示包含词语 T_i 的文档总数。

4.3.3 DBSCAN 聚类算法原理

聚类算法包括比如基于层次的聚类算法 BIRCH，基于网格的聚类算法 STING 基于划分的聚类算法 k-means，基于密度的聚类算法 DBSCAN 等。DBSCAN 和其他的聚类方法相比，可以在有噪音的数据中发现各种形状和各种大小的簇。其核心思想就是以每个数据点为圆心、领域为半径画个圆。定义在这个圆中数据点的个数为密度，从密度点高处把相近的高密度点逐步都连成一片，进而生成各种簇[7]，见图 11。

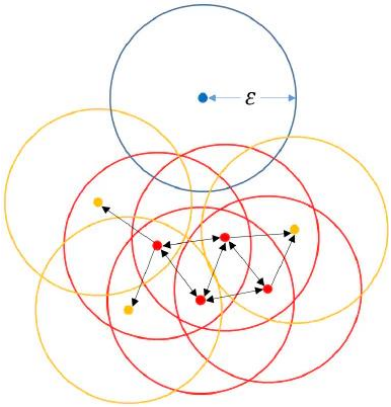

图 11 DBSCAN 聚类原理示意图

4.3.4 挖掘过程及结果

在 Python 的 jieba 库精确模式和全模式两张分词方法中，数据分别包括了问题描述和问题主题两部分内容。因此，我们通过不同的数据得到了不同的结果。经过实验表明，使用留言主题和全模式得到的聚类效果更加明显，具体情况见表 2。

表 2 不同数据的实验结果

	留言主题	留言详情
精确模式	[36, 23, 20, 11, 10]	[20, 9, 8, 8, 7]
全模式	[34, 32, 22, 12, 10]	[29, 25, 15, 15, 7]

再将经过分词提取出来的关键字使用 TF-IDF 算法将每段文字转化成一个向量，从中得到了高维的稀疏矩阵。在此基础上，本文利用 PCA 降维保留了 90%的信息，最后得到（4322，2043）的矩阵。结果见图 12。

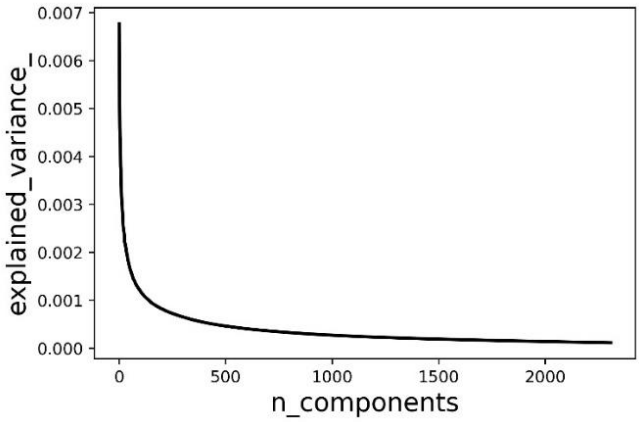

图 12 PCA 降维结果示意图

进一步使用密度聚类方法，在半径 eps 参数的选择上，经过实验我们使用 0.8 作为模型的超参，见表 3 。在聚类过程中，考虑到分组中个数最多的一类评论过多、无中心主题。于是我们选择每类中个数不大于 50 的前五类，作为排名前 5 的热点问题。得到的一组分类结果如图 13。

表 3 不同 eps 半径聚类前五类的个数情况表

eps	0.9	0.8	0.7	0.6	0.5
聚类前五个数	16 11 10 9 9	34 32 22 12 10	23 19 13 10 7	15 13 10 10 5	11 9 5 5 5

	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	kind
584	202249	A00092267	投诉A8县住房公积金无法办理组合贷款	2019/5/10 15:25:47	\n\t\t\t\t\t\n\t\t\t\t\t长金管委（2018）8号、长金管发（2019...	0	0	12
895	209762	A00083931	A市是否可以用住房公积金缴纳房租?	2019/2/27 14:40:21	\n\t\t\t\t\n\t\t\t\t现在像北京、上海等大城市可以用公积金缴纳房租...	0	0	12
1526	224377	A00023049	A7县住房公积金扣缴政策不合理	2019/12/25 15:59:39	\n\t\t\t\t\n\t\t\t\t本人现在名下有一套房无贷款，老公婚前通过公积...	0	0	12
2019	236378	A0007118	询问A市住房公积金贷款的相关问题	2019/3/22 16:44:55	\n\t\t\t\t\n\t\t\t\t我在A市工作了四年，购买了五险一金，现想购买...	0	0	12
2213	241024	A00062451	请问A市住房公积金贷款可以变更姓名吗?	2019/7/12 13:23:52	\n\t\t\t\t\n\t\t\t\t本人32岁，因个人原因想换个姓名，请问现在有...	0	0	12

图 13 部分聚类结果

按照出现次数定义热度指数，那么就可按照热度排名将热点问题筛选出来，热度排名前五的数据见表 4。

表 4 热点排名前五的数据表

热度排名	问题 ID	热度指数	时间范围	地点/人群	问题描述
1	1	34	2019/11/13 至 2020/01/25	A 市 A2 区丽发新城小区	A2 区丽发新城小区附近搅拌站噪音大，污染严重
2	2	32	2019/07/07 至 2019/09/01	A 市伊景园滨河苑	A 市伊景园滨河苑捆绑车位销售
3	3	22	2019/01/12 至 2019/12/15	A7 县、A1 区	多地发生麻将馆扰民问题
4	4	12	2019/1/12	A 市	咨询住房公积金贷款问题
5	5	7	2019/10/15 至 2019/11/27	A 市涉外经济学院	A 市涉外经济学院强制学生实习

4.4 答复意见评价

首先，对留言和答复进行预处理，将数字字母以及标点符号全部去除。在此基础上，对留言和答复进行切片处理，并去除停用词，提取关键字。本文采用余弦相似度作为判断相关性的指标，余弦相似度的原理是通过两个向量之间夹角的余弦值来度量相似性，余弦相似度的值越接近 1 就表示两个向量越相似。定义向量 A 和向量 B 之间余弦相似度为

$$\cos(\theta)=\frac{A\cdot B}{\|A\|\|B\|}=\frac{\sum_{i=1}^{n}A_i\times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\times\sqrt{\sum_{i=1}^{n}(B_i)^2}}$$

根据从留言和答复中提取出来的词，建立了两个向量。具体方法是将两个文本中的词合并在在一起并假设合并之后的长度为 n，根据并集中每个词在两个文本中出现的次数分别赋予两个值,最终形成两个 n 维向量)。计算这两个向量

的之间的余弦相似度，就可以知道在统计学方法中他们的相似度情况。

以计算出来的相似度值的平均值为标准(记为数值 1)，对所有回复进行评判，部分评价结果见表 5。

表 5 部分评价结果表

编号	回复内容
1	您好！您的留言已收悉。现将有关情况回复如下：因您未留下联系方式及投诉的相关证据材料，市工商局无法根据您提供的信息进行投诉信息的登记分送和处理。您可直接拨打我局消费者投诉举报热线 0731-12315 进行反映。感谢您对我们工作的支持、理解与监督！2018 年 12 月 28 日
2	您好！您的留言已收悉。现将有关情况回复如下：因您未留下具体联系方式，市工商局无法与您联系获取详细信息及证据，请您直接拨打市工商局合同监管处电话 0731-0000-00000000 进行反映。感谢您对我们工作的支持、理解与监督！2018 年 12 月 17 日
3	您好！您的留言已收悉。现将有关情况回复如下：对于您的建议，市交通运输局将派遣工作人员现场勘察，进一步调研客流，逐步调整公交站位置。感谢您对我们工作的支持、理解与监督！2018 年 12 月 5 日
4	您好！您的留言已收悉。现将有关情况回复如下：目前，市城乡规划局正在编制《A 市轨道交通线网规划修编规划》，对于您的意见和建议，将充分论证研究。感谢您对我们工作的支持、理解与监督！2018 年 11 月 8 日

五、模型的优化

在问题三中，我们根据余弦相似度的算法已经将相关性不高的回复选取了出来，但其中不乏一些正常的评论被错误归类，仅用相似性就将其纳入低质量不是特别合理。如果我们能在这些相关性很低的回复中选取出“垃圾回复”作为质量很差的回复会更好。为了实现这一目的，我们选用了 python sklearn 机器学习算法系列中的 LogisticRegression(逻辑回归)用以识别“垃圾回复”。首先我们选取了所有相关性低的回复，依然使用 jieba 分词对这些回复进行分词。在得到分词后就要计算每个词的 TF-IDF 权重依此为参数作为算法的特征值，是否为“垃圾回复”为分类类型值。最后，选用 LogisticRegression 逻辑回归进行二分类(分为正常的和质量差的)。

在建立逻辑回归模型时，我们从所有回复中随机选取了 150 个回复作为训练集，并经过人工识别后贴上”正常回复”(用 0 表示)或“垃圾回复”(用 1 表示)的标签，然后进行训练。之后我们把所有低相关性的回复(这里选择的是所有评判值小于 0.5 的回复)作为测试集进行测试。得到的部分结果见图 14。

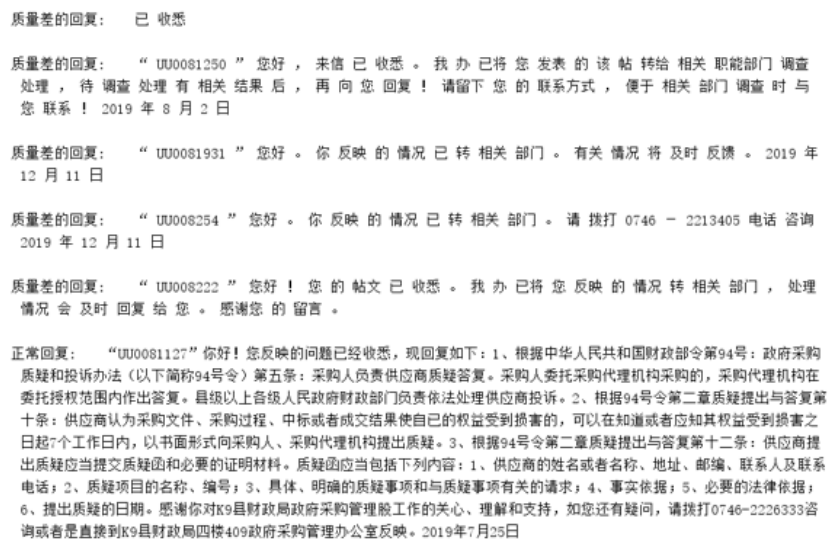

图 14 二分类处理的部分结果

可以看到，根据回复的内容来分析，这种分类的结果是比较令人满意的。所有“垃圾回复”我们都总结在了一张表格当中，部分结果见表 6。

表 6 部分垃圾回复分类表

序号	留言

1	网友“A0009233”，您好，您的留言已收悉，现将具体内容答复如下：关于来信人建议“白竹坡路口”更名为“马坡岭小学”，原“马坡岭小学”取消，保留“马坡岭”的问题。公交站点的设置需要方便周边的市民出行，现有公交线路均使用该三处公交站站名，市民均已熟知，因此不宜变更。感谢来信人对我市公共交通的支持与关心。2019 年 5 月 5 日
2	网友“UU008654”您好！您的留言已收悉。现将有关情况回复如下：目前，市城乡规划局正在编制《A 市轨道交通线网规划修编规划》，对于您的意见和建议，将充分论证研究。感谢您对我们工作的支持、理解与监督！2018 年 10 月 30
3	网友“UU008932”您好！您的留言已收悉。现将有关情况回复如下：对您反映的“举报 A 市 A4 区广胜村前任及现任村支两委的问题”，A4 区纪委区监委进行调查、核实及妥善处置。对于调查处置结果，可直接联系 A4 区纪委区监委办公室获取（电话：0731-0000-00000000）。感谢您对我们工作的支持、理解与监督！2018 年 8 月 16 日
4	网友“UU008957”您好！您的留言已收悉。现将有关情况回复如下：很感谢您的建议，我局经实地勘察后，在下一步线网优化中考虑您的建议，感谢您的来信。感谢您对我们工作的关心、监督与支持。2016 年 12 月 26 日

另外我们从相关性低的回复中随机选取了 50 个回复进行准确度测试，其中 1 代表正常回复，0 代表“垃圾回复”，部分测试结果见表 7，且计算准确率在 92%左右。

表 7 部分测试结果表

	回复类型	预测类型
回复 1	0	1
回复 2	1	1

回复 3	1	1
回复 4	1	1
回复 5	1	1
回复 6	0	0

参考文献

[1] 石凤贵. 基于 TF-IDF 中文文本分类实现[J]. 现代计算机,2020(06):51-54+75.

[2]官琴,邓三鸿,王昊. 中文文本聚类常用停用词表对比研究[J]. 数据分析与知识发现,2017,1(03):72-80

[3]张谦,高章敏,刘嘉勇. 基于 Word2vec 的微博短文本分类研究[J]. 信息网络安全,2017(01):57-62.

[4]李骁,黄征. 基于 GRU 网络的互联网信息挖掘[J]. 信息技术,2018(03):1-5+9.

[5]段丹丹,唐加山,温勇,袁克海.基于 BERT 的中文短文本分类算法的研究[J/OL]. 计算机工程:1-12[2020-05-05].

[6] 江湖人称冷不丁. 通俗理解 TF-IDF[EB/OL]. https://www.jianshu.com/p/0d7b5c226f39，2018.7/2020.5

[7] YeZhu. 基于密度的聚类方法 Density-based clustering[EB/OL]. https://www.jianshu.com/p/0f33ed1c38b9，2017.9/2020.5

附 录

第一问代码：

```
import pandas as pd
from keras import backend as K
import tensorflow as tf
from keras.callbacks import Callback
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score
import matplotlib.pyplot as plt
import pandas as pd
import jieba
import numpy as np
import sklearn as sk
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MultiLabelBinarizer
import os
import codecs
from tensorflow.keras.preprocessing import sequence
from keras_bert import Tokenizer, load_trained_model_from_checkpoint
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Flatten, Embedding, Dropout
from tensorflow.keras import Input, Model, losses, Sequential
from tensorflow.keras.layers import Lambda, Dense, Bidirectional, LSTM, GRU, Conv1D,
Flatten, Dropout
from tensorflow.keras.optimizers import Adam
# 读取数据
df1 = pd.read_excel('./附件 2.xlsx')
df1.head()
# 把标签化为数组
def change2num(x):
    if x == '城乡建设':
        return 0
    if x == '环境保护':
        return 1
    if x == '交通运输':
        return 2
    if x == '教育文体':
        return 3
    if x == '劳动和社会保障':
        return 4
    if x == '商贸旅游':
```

```
            return 5
        if x == '卫生计生':
            return 6
df1['y'] = df1['一级标签'].map(change2num)
# 设置超参
voc_size = 10000
data_num = 10000
syspath = r'D:/Python_first/Tensorflow'
os.chdir(syspath)
# 加载 BERT 的预训练模型
config_path = r'.\chinese_L-12_H-768_A-12\bert_config.json'
checkpoint_path = r'.\chinese_L-12_H-768_A-12\bert_model.ckpt'
dict_path = r'.\chinese_L-12_H-768_A-12\vocab.txt'
# 句子的最大长度
maxlen = 50
# 将 bert 模型中按字进行编码
token_dict = {}
with codecs.open(dict_path, 'r', 'utf-8') as reader:
    for line in reader:
        token = line.strip()
        token_dict[token] = len(token_dict)
class OurTokenizer(Tokenizer):
    '''
    关键在  Tokenizer 这个类，要实现这个类中的方法，其实不实现也是可以的
    目的是 扩充 vocab.txt 文件的
    '''
    def _tokenize(self, text):
        R = []
        for c in text:
            if c in self._token_dict:
                R.append(c)
            elif self._is_space(c):
                R.append('[unused1]')
            else:
                R.append('[UNK]')
        return R
tokenizer = OurTokenizer(token_dict)
all_data = []
# 返回的 x1,是经过编码过后得到，纯整数集合
# 返回的 x2,源码中 segment_ids，表示区分第一句和第二句的位置。
X1 = []
X2 = []
for line in data.X:
    # tokenizer.encode(first, second, maxlen)
```

```
        x1, x2 = tokenizer.encode(first=line)
        X1.append(x1)
        X2.append(x2)
# 利用 Keras 的 API 进行对数据集补齐操作。
X1 = sequence.pad_sequences(X1, maxlen=maxlen, padding='post', truncating='post')
X2 = sequence.pad_sequences(X2, maxlen=maxlen, padding='post', truncating='post')
wordvec = bert_model.predict([X1, X2])
# 加载 Google 训练好的模型 bert
bert_model = load_trained_model_from_checkpoint(config_path, checkpoint_path,
seq_len=None)
def change(x):
        l = []
        l.append(x)
        return l
y1 = data.y.map(change)
y = MultiLabelBinarizer().fit_transform(y1)
X_train, X_test, y_train, y_test = train_test_split(wordvec, y, test_size=0.3)
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.2)
# 写一个 LossHistory 类，保存 loss 和 acc
class LossHistory(keras.callbacks.Callback):
        def on_train_begin(self, logs={}):
                self.losses = {'batch': [], 'epoch': []}
                self.accuracy = {'batch': [], 'epoch': []}
                self.val_loss = {'batch': [], 'epoch': []}
                self.val_acc = {'batch': [], 'epoch': []}
        def on_batch_end(self, batch, logs={}):
                self.losses['batch'].append(logs.get('loss'))
                self.accuracy['batch'].append(logs.get('acc'))
                self.val_loss['batch'].append(logs.get('val_loss'))
                self.val_acc['batch'].append(logs.get('val_acc'))
        def on_epoch_end(self, batch, logs={}):
                self.losses['epoch'].append(logs.get('loss'))
                self.accuracy['epoch'].append(logs.get('acc'))
                self.val_loss['epoch'].append(logs.get('val_loss'))
                self.val_acc['epoch'].append(logs.get('val_acc'))
# 实例化 history
history = LossHistory()
# 显示 F1 score 的值
class Metrics(Callback):
        def __init__(self, val_data, val_label):
                super(Callback, self).__init__()
                self.val_data = val_data
                self.val_label = val_label
        def on_train_begin(self, logs={}):
```

```
        self.val_f1s = []
    def on_epoch_end(self, epoch, logs={}):
        val_predict = (np.asarray(self.model.predict(self.val_data))).round()
        val_targ = self.val_label
        _val_f1 = f1_score(val_targ, val_predict, average='micro')
        self.val_f1s.append(_val_f1)
        print("— val_f1: %f " % _val_f1)
        return
# 实例化 metrics
metrics = Metrics(X_test, y_test)
# fine tuning 部分
model = Sequential()
model.add(Bidirectional(LSTM(128, dropout=0.1)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(7, activation='softmax'))
model.compile(loss=losses.categorical_crossentropy, optimizer=Adam(1e-5), metrics=['acc'])
model = Sequential()
model.add(Bidirectional(LSTM(128, dropout=0.1)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(7, activation='softmax'))
model.compile(loss=losses.categorical_crossentropy, optimizer=Adam(1e-5), metrics=['acc'])
# 训练
model.fit(X_train, y_train, batch_size=32, epochs=50, verbose=2, callbacks=[history, metrics],
          validation_data=(X_test, y_test))
# 画模型在 train 和 test 上的 acc
train_loss = history.losses['epoch']
train_acc = history.accuracy['epoch']
val_loss = history.val_loss['epoch']
val_acc = history.val_acc['epoch']
iters = range(len(train_loss))
plt.figure()
# acc
plt.plot(iters, train_acc, 'r', label='train acc')
# val_acc
plt.plot(iters, val_acc, 'b', label='val acc')
plt.grid(True)
plt.xlabel('epoch')
plt.ylabel('acc')
plt.legend(loc="lower right")
```

```
plt.savefig(r'D:\Python_first\学习内容\数学建模 3\acc3.jpg', dpi=400, bbox_inches='tight')
plt.show()
```

检测 models 代码：

```
import pandas as pd
import tensorflow as tf
from keras.callbacks import Callback
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score
import matplotlib.pyplot as plt
import pandas as pd
import jieba
import numpy as np
import sklearn as sk
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MultiLabelBinarizer
from tensorflow.keras.preprocessing import sequence
from keras_bert import Tokenizer, load_trained_model_from_checkpoint
from tensorflow.keras.optimizers import Adam
# 读取数据
df1 = pd.read_excel('D:\Python_first\学习内容\数学建模 3\附件 2.xlsx')
df1.head()
# 把标签化为数组
def change2num(x):
    if x == '城乡建设':
        return 0
    if x == '环境保护':
        return 1
    if x == '交通运输':
        return 2
    if x == '教育文体':
        return 3
    if x == '劳动和社会保障':
        return 4
    if x == '商贸旅游':
        return 5
    if x == '卫生计生':
        return 6
df1['y'] = df1['一级标签'].map(change2num)
data = df1[3000: 3100]
# 超参
voc_size = 10000
data_num = 10000
syspath = r'D:/Python_first/Tensorflow'
os.chdir(syspath)
```

```
# 调用 Bert-Pre-training
config_path = r'D:\Python_first\Tensorflow\chinese_L-12_H-768_A-12\bert_config.json'  # 加载配置文件
checkpoint_path = r'D:\Python_first\Tensorflow\chinese_L-12_H-768_A-12\bert_model.ckpt'
dict_path = r'D:\Python_first\Tensorflow\chinese_L-12_H-768_A-12\vocab.txt'
maxlen = 50  # 句子的最大长度，padding 要用的
# 将 bert 模型中的 字 进行编码
token_dict = {}
with codecs.open(dict_path, 'r', 'utf-8') as reader:
    for line in reader:
        token = line.strip()
        token_dict[token] = len(token_dict)
class OurTokenizer(Tokenizer):
    '''
    关键在  Tokenizer 这个类，要实现这个类中的方法，其实不实现也是可以的
    目的是 扩充 vocab.txt 文件的
    '''
    def _tokenize(self, text):
        R = []
        for c in text:
            if c in self._token_dict:
                R.append(c)
            elif self._is_space(c):
                R.append('[unused1]')
            else:
                R.append('[UNK]')
        return R
tokenizer = OurTokenizer(token_dict)
all_data = []
X1 = []
X2 = []
for line in data:
    x1, x2 = tokenizer.encode(first=line)
    X1.append(x1)
    X2.append(x2)
# 利用 Keras API 进行对数据集  补齐  操作。
X1 = sequence.pad_sequences(X1, maxlen=maxlen, padding='post', truncating='post')
X2 = sequence.pad_sequences(X2, maxlen=maxlen, padding='post', truncating='post')
'''
:param X1:经过编码过后的集合
:param X2:经过编码过后的位置集合
:return:模型
'''
bert_model = load_trained_model_from_checkpoint(config_path, checkpoint_path,
```

```
seq_len=None)
wordvec = bert_model.predict([X1, X2])
def change(x):
    l = []
    l.append(x)
    return l
y1 = data.y.map(change)
y = MultiLabelBinarizer().fit_transform(y1)
# 调用模型
model = keras.models.load_model('./best.h5')
# 查看结果
result = model.predict_classes(wordvec)
```

第二问代码：

```
import pandas as pd
import re
import jieba
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn import decomposition
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
# 读取数据
df1 = pd.read_excel('./附件 3.xlsx')
df1 = df1[pd.to_datetime(df1['留言时间']) > pd.to_datetime('2019-01-01')]
stops = [line.strip() for line in open('./stopwords.txt', encoding='UTF-8').readlines()]
stops.extend(['市', '区', '山', '路', '请', '恳请', '希望', '郡', '县', '街', ])
for i in range(10):
    stops.remove(str(i))
# 清洗文本
def clearTxt(line: str):
    if (line != ''):
        line = line.strip()
        # 去除文本中的英文和数字
        #             line = re.sub("[a-zA-Z0-9]", "", line)
        # 去除文本中的中文符号和英文符号
        line = re.sub("[\s+\.\!\/_,$%^*(+\"\'；：“”．]+|[+——！，。？?、~@#
￥%……&*（）]+", "", line)
        return line
    return None
# 文本切割
def sent2word(line):
    segList = jieba.cut(line, cut_all=True)
```

```
        segSentence = ''
        for word in segList:
            if word != '\t':
                segSentence += word + " "
        return segSentence.strip().split(' ')
df2 = df1['留言主题'].map(clearTxt)
# 分词 去停用词
df3 = []
for i in df2:
        sen = sent2word(i)
        new = []
        for j in sen:
            if j not in stops:
                new.append(j)
        df3.append(' '.join(new))
# 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(df3))
# 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()
# PCA 降维过程
pca = decomposition.PCA(n_components=0.9)
pca_data = pca.transform(tfidf_matrix)
plt.figure()
plt.plot(pca.explained_variance_, 'k', linewidth=2)
plt.xlabel('n_components', fontsize=16)
plt.ylabel('explained_variance_', fontsize=16)
plt.savefig(r'D:\Python_first\学习内容\数学建模 3\pca.jpg', dpi=400, bbox_inches='tight')
plt.show()
#使用密度聚类方法
clf = KMeans(n_clusters=20)
dbscan = DBSCAN(eps=0.8, min_samples=5)
# clf.fit(tfidf_matrix)
res = dbscan.fit_predict(pca_data)
df1['kind'] = res
df1.groupby('kind').count().sort_values('留言编号', ascending=False)[:8]
# 保存数据
df1[df1['kind'] == 10].to_csv('./分类五.csv')
```

第三问代码：

```
# coding=utf-8
import xlrd
import xlwt
from xlutils.copy import copy
#  正则包
import re
# html  包
import html
#  自然语言处理包
import jieba
import jieba.analyse
#  机器学习包
from sklearn.metrics.pairwise import cosine_similarity
class CosineSimilarity(object):
    """
    余弦相似度
    """
    def __init__(self, content_x1, content_y2):
        self.s1 = content_x1
        self.s2 = content_y2
    @staticmethod
    def extract_keyword(content):   #  提取关键词
        #  正则过滤 html  标签
        re_exp = re.compile(r'(<style>.*?</style>)|(<[^>]+>)', re.S)
        content = re_exp.sub(' ', content)
        # html  转义符实体化
        content = html.unescape(content)
        #  切割
        seg = [i for i in jieba.cut(content, cut_all=True) if i != '']
        #  提取关键词
        keywords = jieba.analyse.extract_tags("|".join(seg), topK=200, withWeight=False)
        return keywords
    @staticmethod
    def one_hot(word_dict, keywords):   # oneHot 编码
        # cut_code = [word_dict[word] for word in keywords]
        cut_code = [0]*len(word_dict)
        for word in keywords:
            cut_code[word_dict[word]] += 1
        return cut_code
    def main(self):
        #  去除停用词
jieba.analyse.set_stop_words('C:/Users/12177/Desktop/stopwords.txt')

        #  提取关键词
```

```
            keywords1 = self.extract_keyword(self.s1)
            keywords2 = self.extract_keyword(self.s2)
            # 词的并集
            union = set(keywords1).union(set(keywords2))
            # 编码
            word_dict = {}
            i = 0
            for word in union:
                word_dict[word] = i
                i += 1
            # oneHot 编码
            s1_cut_code = self.one_hot(word_dict, keywords1)
            s2_cut_code = self.one_hot(word_dict, keywords2)
            # 余弦相似度计算
            sample = [s1_cut_code, s2_cut_code]
            # 除零处理
            try:
                sim = cosine_similarity(sample)
                return sim[1][0]
            except Exception as e:
                print(e)
                return 0.0
# 测试
numb = list()
if __name__ == '__main__':
    data = xlrd.open_workbook('C:/Users/12177/Desktop/附件 4.xlsx')
    table = data.sheet_by_index(0)
    for i in range(1,table.nrows):
        content_x = table.cell(i, 4).value
        content_y = table.cell(i, 5).value
        content_x = re.sub("[a-zA-Z0-9]", "", content_x)
        # 去除文本中的中文符号和英文符号
        content_x = re.sub("[\s+\.\!\/_,$%^*(+\"\'；：“”．]+|[+——！，。？?、~@#
￥%……&*（）]+", "", content_x)
        content_y = re.sub("[a-zA-Z0-9]", "", content_y)
        # 去除文本中的中文符号和英文符号
        content_y = re.sub("[\s+\.\!\/_,$%^*(+\"\'；：“”．]+|[+——！，。？?、~@#
￥%……&*（）]+", "", content_y)
        similarity = CosineSimilarity(content_x, content_y)
        similarity = similarity.main()
        numb.append(similarity)
        # fp.writelines(str(similarity)+'\n')
        # print('相似度: %.2f%%' % (similarity*100))
    with open('C:/Users/12177/Desktop/相似度.txt', 'w', encoding='utf-8') as fp:
```

```
        sum_num = 0
        for i in numb:
            sum_num = sum_num+i
        ave_num = sum_num/len(numb)
        for i in range(len(numb)):
            numb[i] = numb[i]/ave_num
            fp.writelines(str(numb[i])+'\n')
```

优化后的代码：

```
# coding=utf-8
#  总体思路就是:分词-----计算 TF-IDF 权重-----选用模型预测
import warnings
import numpy as np
import matplotlib.pyplot as plt
from sklearn.externals import joblib
import pandas as pd
import matplotlib as mpl
from sklearn import metrics
import jieba.posseg as pseg
from sklearn.model_selection    import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model.logistic import LogisticRegression
#  忽略一些版本不兼容等警告
warnings.filterwarnings("ignore")
#  从磁盘读取 y 原始数据进行训练
X = []
Y1 = []
Y2 = []
f = open('train_data.txt',encoding='utf-8')
for v in f:
    X.append([v.strip('\n').split('/')[0], v.strip('\n').split('/')[1]])    # strip('\n')是去除换行符\n
f.close()
#  进行分词，分词后保存在 Y 中
for i in range(len(X)):
    words = pseg.cut(X[i][1])
    str1 = ""
    for key in words:
        str1 += key.word
        str1 += ' '
    Y1.append(str1)   #  回复内容
    Y2.append(X[i][0])   #  是否是垃圾的标志
#  将样本分为训练集和测试集
x_train_Chinese, x_test_Chinese, y_train, y_test = train_test_split(Y1, Y2, train_size=0.99)
```

```
# 通过 TfidfVectorizer 算出 TF-IDF 权重
vectorizer = TfidfVectorizer()
x_train = vectorizer.fit_transform(x_train_Chinese)
'''
#模块一:测试准确率，召回率等信息表
#核心代码
classifier=LogisticRegression()
classifier.fit(x_train,y_train)
y_tanin_predict=classifier.predict(x_train)
print(metrics.classification_report(y_train,y_tanin_predict))          #包含准确率，召回率等信息表
print(metrics.confusion_matrix(y_train,y_tanin_predict))                #混淆矩阵
'''
# 模块二:预测信息
# 读取待预测的回复读取到 X1 中
X1 = []
X2 = []
f = open('demand prediction.txt',encoding='utf-8')
for v in f:
    X1.append(v.strip('\n'))
f.close()
# 进行分词，分词后保存在 X2 中
for i in range(len(X1)):
    words = pseg.cut(X1[i])
    str1 = ""
    for key in words:
        str1 += key.word
        str1 += ' '
    X2.append(str1)  # 回复内容
# 计算待预测回复的 TF-IDF 权重
x_demand_prediction = vectorizer.transform(X2)
# 预测
classifier = LogisticRegression()
classifier.fit(x_train, y_train)
y_predict = classifier.predict(x_demand_prediction)
# 输出
print('---------------------------------------回复预测结果---------------------------------------')
with open('cdhf.txt','w',encoding='utf-8') as fp:
    for i in range(len(X1)):
        if int(y_predict[i]) == 0:
            # continue
            print('正常回复:\t' + X1[i] + '\n')
        else:
```

```
print('质量差的回复:\t' + X2[i] + '\n')
# fp.writelines(X2[i]+'\n')
```