“智慧政务”中的文本挖掘应用

摘 要

伴随着互联网+的飞速发展，社交媒体渐渐改变了社会中人们的互动方式。越来越多的人通过微信，微博，官网等网络平台进行留言，有关社情民意的留言数量也节节攀升，有效分类以及整理此类留言能提高相关部门的管理水平和工作效率。

针对问题一，首先进行数据清洗，去除附件二留言中的空格和x序列；再使用结巴分词，引进停用词表去掉常用词，解决文本无意义表达太多的问题，采用$TF-IDF$权值向量方法，将本文数据转化成文本特征向量，同时选择特征词，绘制词云（见图一）。完成数据预处理的工作后，本文采用了递归神经网络模型和SVM模型对附件2的留言详情进行分类，算出两者F_1值分别为0.83和0.92。可知基于SVM的文本分类器效果更好，故采用SVM建立一级标签分类模型。

针对问题二，按照相同的步骤对数据进行预处理，同样地，采用$TF-IDF$权值向量方法对文本进行转化并提取特征值。为找到相关热点问题，采用$K-Means$聚类的方法将文本分为六类，再采用选定的热度评价指标公式:$y=\left[\left(\frac{a}{a_n}+\frac{b}{b_n}+\frac{n}{N}\right)\div 3\right]\times 1000, a,b,n$表示此类所获点赞数，反对数，总点数,$a_n,b_n,c_n$代表所有留言文本总赞数，总反对数，以及总点数。选出热度排名前五的五个热点问题，从高到低依次是：油烟扰民污染空气；搅拌厂噪音扰民灰尘污染；开发商违规销售涨价；相关政府不同意拆迁；拆迁停滞村民居无定所。并给出文件“留言问题表.xls”和“热点问题留言明细表.xls”。

针对问题三，将完整性，相关性，可解释性分成三个环节对答复意见进行评分。基于向量空间模型计算答复留言的完整性，相关性则通过深度学习方法进行计算，得到每个答复意见相关性数值，完整性依据深度神经网络计算，将以上三类数值按照以下标准取值给分：数值低于 0.5 给零分，(0.5,0.6]区间给 1 分，在(0.6,0.7]之间给 2 分，在(0.7,0.8]之间给 3 分，(0.8,0.9]之间给 4 分，(0.9,1.0]之间给 5 分。答复意见最终质量得分为完整性得分，相关性得分，以及可解释性得分之和。总分为 15 分，根据答复意见总分进行评价：9 分以下此答复评定为不合格，9 至 12 分为良好，12 分以上为优秀。

关键词：svm模型，$TF-IDF$权值向量，文本分类，$K-Means$聚类

Abstract

With the rapid development of Internet +, social media has gradually changed the way people interact in society. More and more people leave messages through Wechat, Weibo, official website and other online platforms, and the number of messages about social situation and public opinion is also rising. Effective classification and arrangement of such messages can effectively improve the management level and work efficiency of the relevant departments.

Aiming at the first problem, we first clean the data to remove the spaces and x-sequences in the messages in Annex II; then we use stuttering participle and stop the word list to remove the common words to solve the problem of too much meaningless expression of the text. We use the TF-IDF weight vector method to transform the data into text feature vectors, and at the same time select feature words to draw word clouds (see figure). After completing the data preprocessing, this paper uses recurrent neural network model and SVM model to classify the message details of attachment 2, and calculates that their F1 values are 0.83and 0.92respectively. It is known that the effect of text classifier based on SVM is better, so SVM is used to establish a first-level label classification model.

Aiming at the second problem, the data is preprocessed according to the same steps, and similarly, the TF-IDF weight vector method is used to transform the text and extract the eigenvalues. In order to find the relevant hot issues, the text is divided into six categories by K-Means clustering method, and then the selected heat evaluation index formula is used: $y = \left[(\frac{a}{a_n} + \frac{b}{b_n} + \frac{n}{N}) \div 3\right] \times 1000$, a, b, nrepresent the number of thumb up, counter number, and total points gained for this class, a_n, b_n, c_nrepresent the total likes, total objections, and total points of all message text Select the top five hot issues, from high to low: Soot disturbs and pollutes the air.Noise from mixing plant disturbs people dust pollution, Developers mis-sell and raise prices, The government did not agree to the demolition, The villagers have no fixed place to live .And give the list of questions on messages (see table) and the list of messages on hot issues (see table).

In view of question 3, the completeness, relevance and interpretability are divided into three links to grade the response messages. The integrity of the reply message is calculated based on the vector space model, and the correlation is calculated by the deep learning method, and the correlation value of each reply opinion is obtained. The above three types of values are scored according to the following criteria: values less than 0.5 to zero, 1 point in the range of (0.5 ~ 0.6), 2 points between (0.6 ~ 0.7], and 3 points between (0.7 ~ 0.8]. Give 4 points between (0.8 and 0.9) and 5 points between (0.9) and 1.0]. The final quality of the response is divided into the sum of integrity score, relevance score, and explanatory score. The total score is 15, which is evaluated according to the total score of the reply: below 9 points, this reply message is unqualified, 9 to 12 points are good, and more than 12 points are excellent.

Keywords：SVM model, tf-idf weight vector, text classification, k-means clustering

目 录

一、问题重述

对于问题一，群众留言信息会被相关工作人员按照一定划分标准进行分类，然后移交到相应部门解决。附件一给出一、二、三级分类标签体系，以及附件二的群众留言信息，要求基于附件二建立与留言内容有关的一级分类模型。并使用$F-score$评价分类方法。

对于问题二，热点问题指的是某一时间段内，群众最多反映的一类问题。热点问题的及时发现对有关部门问题处理效率的提高有很大的帮助。需要将附件三中某一时间段内，所反映的特定地点或者特定人群的留言分成五类，并且给出热度评价指标,将排名前五的热点问题保存为表一：热点问题表，将热点问题所对应的留言信息，保存为表二：热点问题留言明细表。

对于问题三，要求从“相关性”“完整性”“可解释性”等角度对附件四中留言的答复意见的质量进行评价，同时给出一套可行较准确的评价方案，并试着实现。

二、数据预处理

文本数据预处理是指将特定的信息抽取出来，并且将它们转化成行记录的一系列过程。文本数据预处理的步骤见下图：

图一 数据预处理

首先，基于文本中有大量无关信息，需要有效去除x序列、空格等，x序列包含特殊字符、数字等。

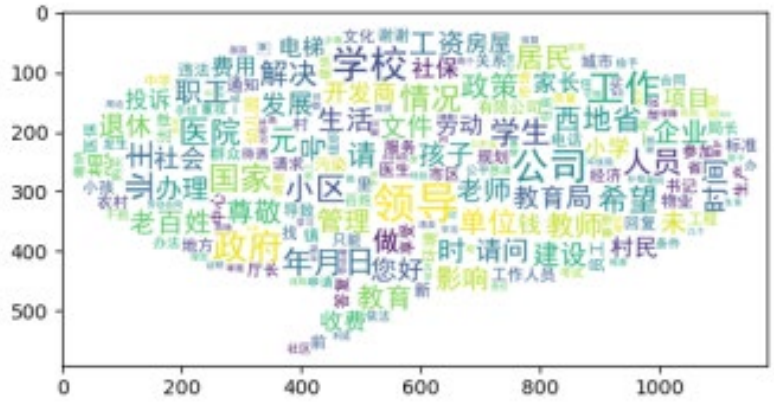
接着采用结巴分词进行分词，为了提高计算的效率以及分类的准确度，需要引

进停用词表将“你好”“谢谢”“这个”等这些不含有效信息的常用词去掉。

分词结束后，还需要根据分词所得到的词袋模型对文本文档做一些特征工程，从而得到计算机可以计算的数据类型。这里本文使用$TF-IDF$算法计算文本的TF正向词频以及IDF逆向逆文本词频，再基于卡方统计方法逆向排列并选取特征词，最后计算每条文本特征词的$TF-IDF$值进行向量化处理。

三、群众留言分类

为了更好的建立关于一级标签的分类模型，在完成数据预处理的工作后，将文本数据通过$TF-IDF$算法转化为向量，绘制图云见图二，



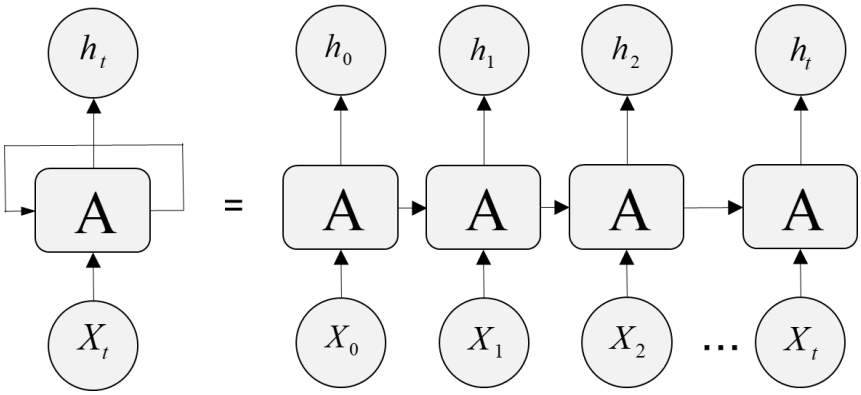
图二 词云图

接下来，本文分别采用了神经网络算法以及SVM模型进行分类，比较F_1值并选取最优模型。

3.1 基于递归神经网络建立一级标签分类模型

3.1.1 递归神经网络原理

神经网络是模拟人大脑神经网络从而处理信息单元的系统。传统的神经网络系统有一个缺点就是它无法长时间记住信息。假如你在看一个故事，想要对每个章节的发生的事情进行分类。然而目前仍然不清楚如何用传统的神经网络算法通过前面故事来推测后面内容。“$LSTM$”递归神经网络的诞生解决了这个问题，其学习序列数据具有长期的依赖性。将递归神经网络展开得到的基本结构如下：

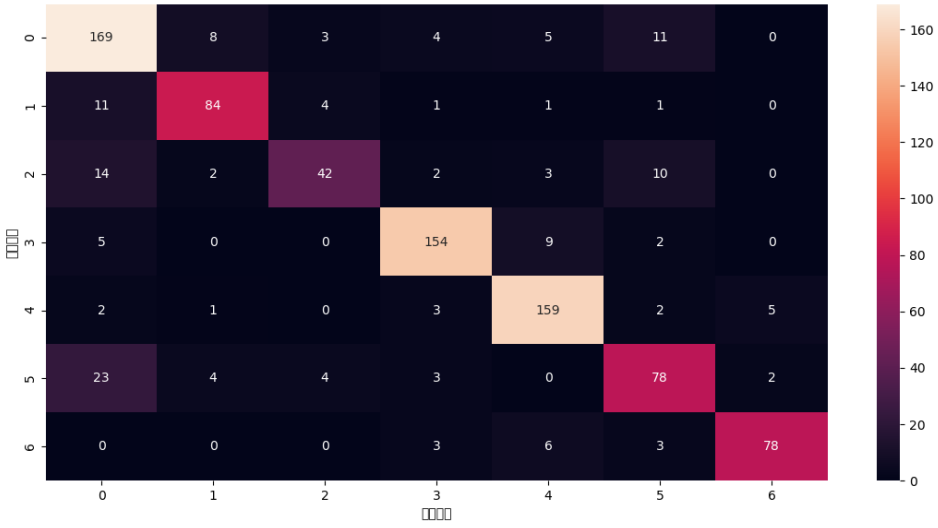


图三 展开的递归神经网络

由图可以看到，当传统神经网络被扩展开来，它每一层都享有相同的权重。*LSTM*之后被证明其比传统的神经网络模型更加的有效，特别是当每个单位时间内有若干层时，整个系统能将其转化为拥有完全一致性的一系列字符。

3.1.2 “*LSTM*”分类结果

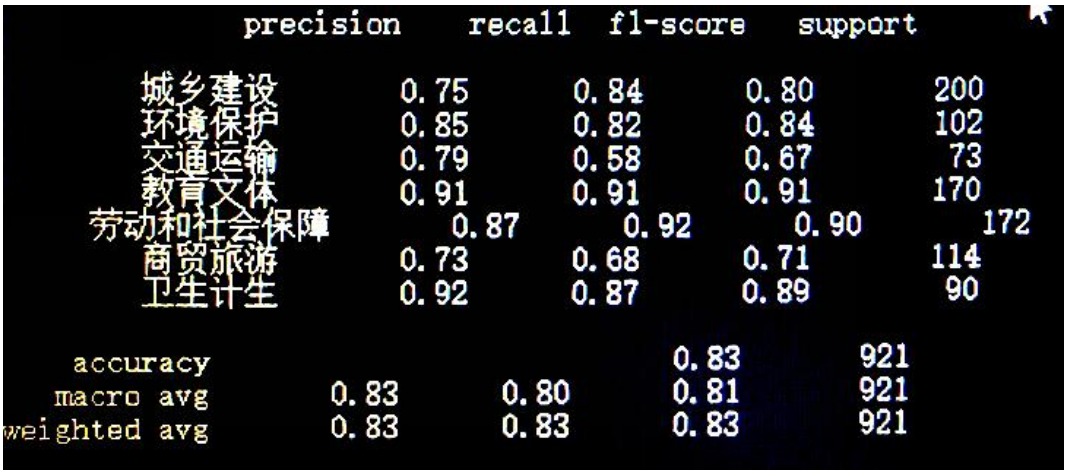
将已经完成预处理，转化为向量值的文本中抽取百分之八十作为测试值训练分类器，剩下百分之二十进行分类，通过递归神经网络模型得到的分类结果。得到的混淆矩阵如下图：



图四 *LSTM*模型分类得到的混淆矩阵

此图中，横坐标为“*Predicted*”，纵坐标为“*Actual*”。观察可见，该混淆矩阵对角线上的留言文本都是预测情况等同于真实情况，即它们都为分类正确的文本。

进一步计算$F-score$，通过$Python$编程得到如下结果：



	precision	recall	f1-score	support
城乡建设	0.75	0.84	0.80	200
环境保护	0.85	0.82	0.84	102
交通运输	0.79	0.58	0.67	73
教育文体	0.91	0.91	0.91	170
劳动和社会保障	0.87	0.92	0.90	172
商贸旅游	0.73	0.68	0.71	114
卫生计生	0.92	0.87	0.89	90
accuracy			0.83	921
macro avg	0.83	0.80	0.81	921
weighted avg	0.83	0.83	0.83	921

图五 $LSTM$模型F_1值

由图可知，第一列数值为每一类的$Precision$值，第二列数据为每一类的$recall$值，通过公式计算得到$LSTM$模型$F_1=0.83$。

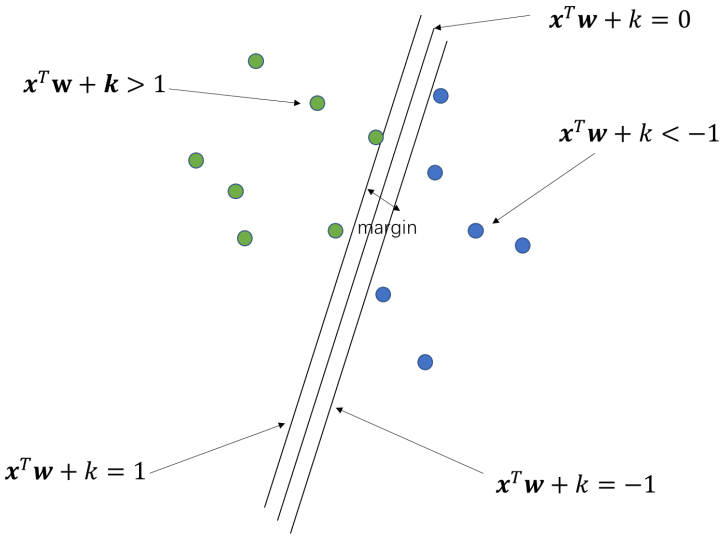
3.2 基于SVM建立一级标签分类模型

3.2.1 SVM模型的原理

支持向量机最早是由$Vapnik$提出的，它是一种能够监督机器学习的算法。支持向量机分为线性可分，与线性不可分的SVM模型。

（1）线性可分的支持向量机模型[1]

如图所示：



图六 线性可分的支持向量机模型

训练样本中$M = (x_1, y_1), (x_2, y_2), \ldots (x_n, y_n), y$的取值从-1 到 1。这里要找到一条直线，其函数表示为：

$$y(x) = w^T x + k。(3\text{-}1)$$

假如该直线具有将训练样本进行正确分类的能力，即$(x_n, y_n) \in M$，就有

$$\begin{cases} wx_n + k \geq 1, y_n = 1 \\ wx_n + k \leq -1, y_n = -1 \end{cases} (3\text{-}2)$$

通过袋鼠变换，进而求得间距为：$Width = \frac{2}{\|w\|} (3\text{-}3)$

在二维空间下，为了是要使距离最大，即：

$$max \frac{2}{\|w\|} = min \frac{1}{2} \|w\|^2 (3\text{-}4)$$

$$s.t wx_n + k \geq 1, n = 1,2,\ldots,m$$

此为，支持向量机的基本公式。接着引入拉格朗日函数，将上面的优化目标函数转化成为没有约束的优化函数：

$$L(w, k, t) - \frac{1}{2} \|w\|^2 - \sum_{n=1}^{m} t_n [y_n (w^T x + k) - 1] (3\text{-}5)$$

其中t_n大于零，为拉格朗日乘子。解出w, k, t后，可得到模型：

$$f(x) = w^T + k = \sum_{n=1}^{m} t_n y_n x_n^T x + k (3\text{-}6)$$

$$s.t \begin{cases} t_n > 0 \\ y_n f(x_n) - 1 \geq 0 \\ t_n (y_n f(x_n) - 1) = 0 \end{cases}$$

(2)线性不可分SVM

当遇到线性不可分的情况时，需要引入δ松弛变量，它用来度量理想的最优超平面与度量数据点之间的关系。同时引入C惩罚参数，用来惩罚放生错误的分类，

当C值越大，其惩罚力度也越大。基于此，可将问题转化为而桂花问题，相同地，引入朗格朗日函数，得到如下函数：

$$D(w,k,\delta,t,u)=\frac{1}{2}\|w\|^2+c\sum_{n=1}^{m}m\,\delta_i-\sum_{n=1}^{m}t_n[y_n(w^Tx+k)-1+\delta_i]-\sum_{n=1}^{i}u_i\delta_i \quad (3\text{-}7)$$

其中u_i,δ_i都大于零为朗格朗日乘子。令$D(w,k,\delta,t,u)$对w,k,δ的偏导函数为零，带入原式，得到对偶问题，有：

$$max=\sum_{n=1}^{m}t_n-\frac{1}{2}\sum_{n=1}^{m}\sum_{j=1}^{m}t_nt_jy_ny_jx_n^Tx_j \quad (3\text{-}8)$$

$$s.t\sum_{n=1}^{m}t_ny_n=0,0\le t_n\le c,n=1,2,\ldots,m$$

为了解决不可分问题通过映射到高纬度空间，可能产生的向量内积维度计算灾难，引入了核函数简化运算来解决这个问题。

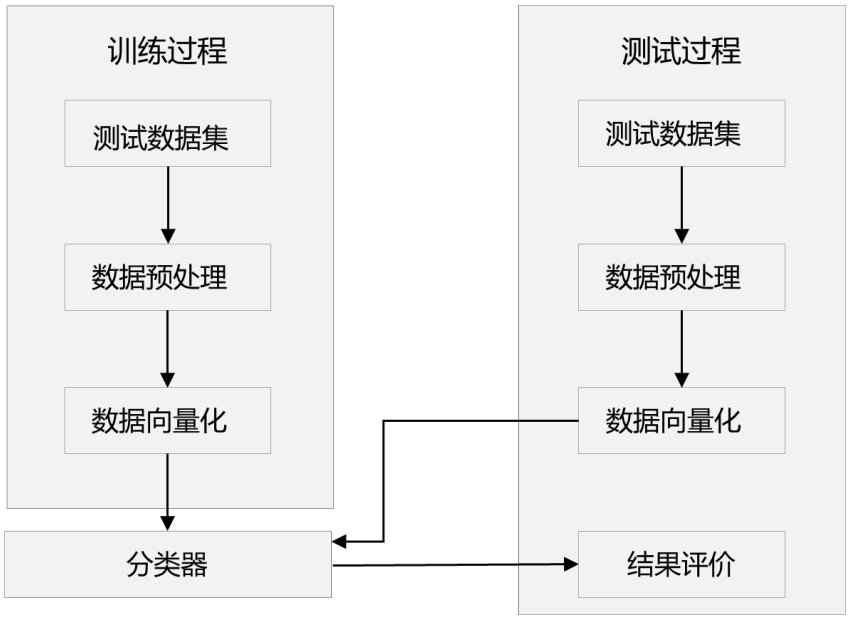
常见的核函数如下表：

表一 常见核函数

名称	表达式	相关参数
线性核	$K(x_i,\ x_j)=x_i{}^Tx_j$	
多项式核	$K(x_i,\ x_j)=(x_i{}^Tx_j)^d$	$d\ge1$为多项式次数
高斯核	$K(x_i,\ x_j)=exp\left(-\frac{\|x_i-x_j\|^2}{2\sigma^2}\right)$	$\sigma>0$为高斯核带宽
拉普拉斯核	$K(\boldsymbol{x_i},\ \boldsymbol{x_j})=exp\left(-\frac{\|x_i-x_j\|}{2\sigma^2}\right)$	$\sigma>0$
Sigmoid 核	$K(x_i,\ x_j)=tan\,h(\beta x_i^Tx_j+\theta)$	$tan\,h$为双曲正切函数，$\beta>0$，$\theta<0$

3. 2. 2*SVM*模型分类结果

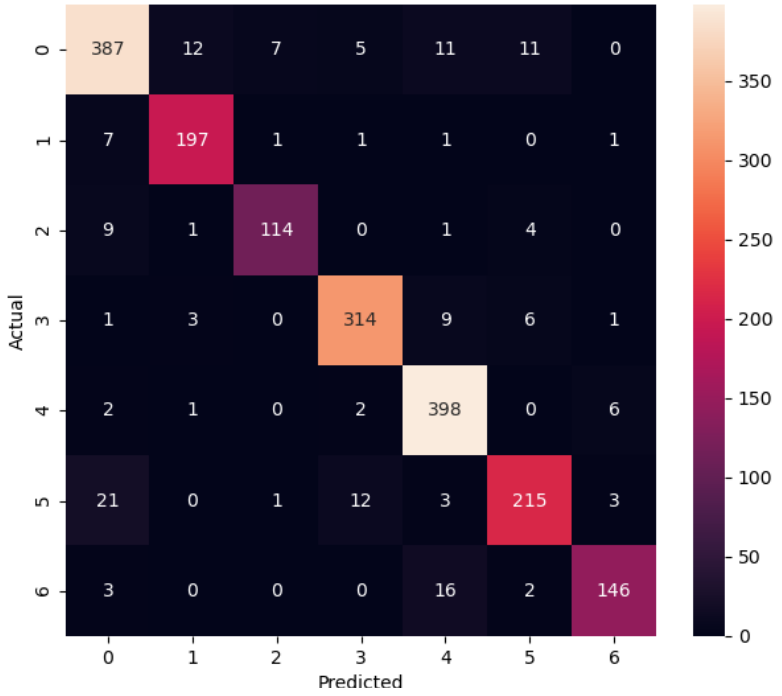
本文首先将百分之七十九的数据做为训练集去训练向量机分类器，将剩下百本之二十一的数据作为测试集进行分类。具体步骤[2]可见下图：



图七 SVM模型分类步骤

由图可以看到，不管是测试集还是训练集都需要将数据进行预处理，并且向量化。通过训练分类器进行调整从而选择最合适的参数与数据集，确保在分类过程中得到最好的结果和最理想的F_1值。

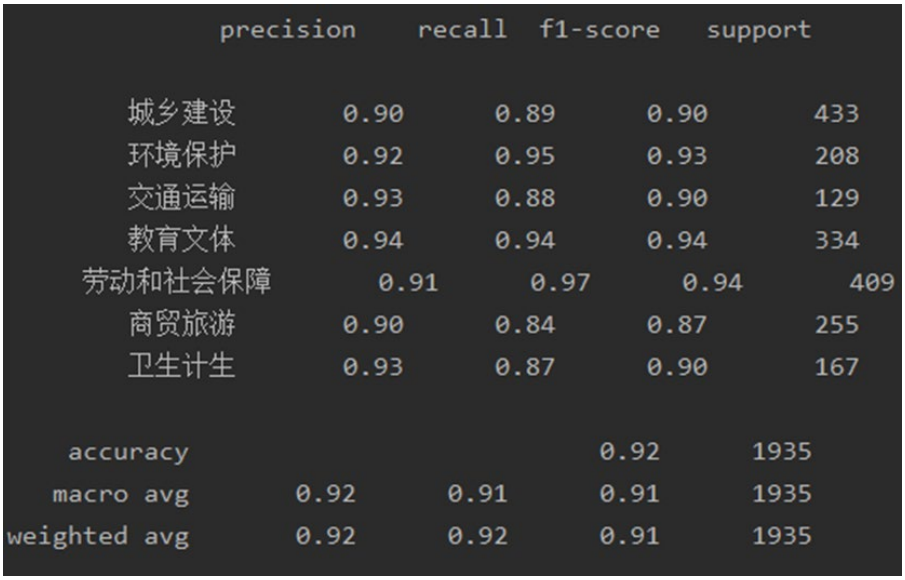
经过训练过程和测试过程，通过代码运行最后得到的混淆矩阵如下：



图八 混淆矩阵

可知，图中对角线上的文本都是$Predicted = Actual$，即预测集同真是集一样，

是正确分类的文本。进而计算F_1值，得到下图：



	precision	recall	f1-score	support
城乡建设	0.90	0.89	0.90	433
环境保护	0.92	0.95	0.93	208
交通运输	0.93	0.88	0.90	129
教育文体	0.94	0.94	0.94	334
劳动和社会保障	0.91	0.97	0.94	409
商贸旅游	0.90	0.84	0.87	255
卫生计生	0.93	0.87	0.90	167
accuracy			0.92	1935
macro avg	0.92	0.91	0.91	1935
weighted avg	0.92	0.92	0.91	1935

图九 运行结果

由图可知总的$Precision$值$= 0.92$与$recall$值$= 0.92$，进而计算得到$F_1 = 0.92$。

比较两个模型的$F - score$可知，训练支持向量机分类器进行分类的效果远好于基于递归神经网络建立的分类模型。故为了使分类精度更高，效果更好，采用基于SVM建立一级标签的分类模型。

四、热点问题挖掘

通过研究本文发现，文本聚类能够用来发现热点问题，同时通过热度计算公式，计算热度进行排名，能够有效找到排名前五的热点问题。本文采用$K - Means$聚类算法将附件三的所有留言文本进行聚类分析。

4.1 $K - Means$聚类算法原理

$K - Means$聚类由$Macqueen$于1967年首次提出，它又称为快速聚类方法，是自下而上的聚类，有着速度较快，易理解，且可以并行计算的优点。

这里，首先给出样本合集：$X = \{X_1, X_2, X_3, \dots, X_n\}$，$n$为样本总数。

聚类的初始中心点有m个$\{X_1, X_2, X_3, \dots, X_m\}$, n_k为第k类样本数：

$$Z_k = \frac{1}{n_k} \sum_{x \in a_k} x \quad (4\text{-}1)$$

接着，定义m个类别为$\{a_1, a_2, a_3, \dots, a_m\}$;

最后，定义目标收敛函数：$E = \sum_k^m \sum_{i=1}^n d_{ki}(Z_k, X_i)$（4-2）

以上为$K-Means$[3]聚类步骤中将要使用的定义

$K-Means$聚类算法的通常步骤如下：

（1） 随机取m个样本点作为每一类初始聚类的中心，记为$\{X_1, X_2, X_3, \ldots, X_m\}$。

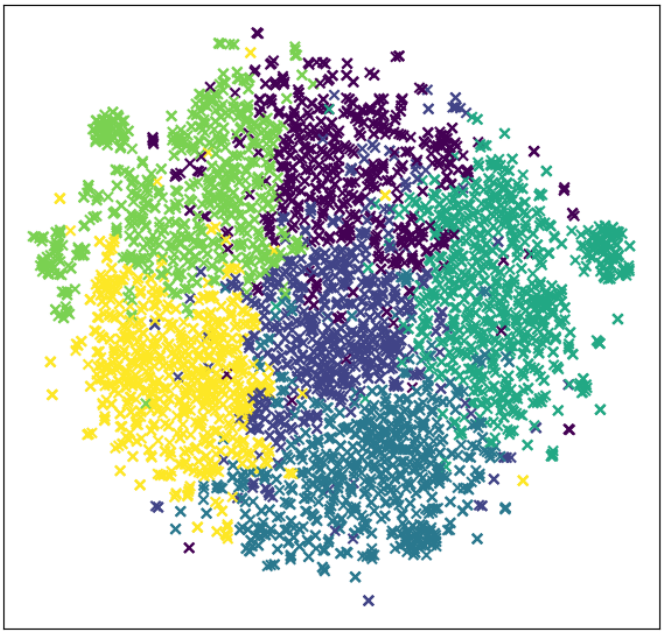
（2） 计算每个尚未分类的样本到各类初始中心点的距离，将它们放到距离最近的初始中心点那一类去。

（3） 用（4-1）式重新计算各类的聚类中心点，记为$\{y(z)_1, y(z)_{2\ldots} y(z)_k\}$，其中$z$代表迭代次数。

（4） 重复执行（2）（3）问，直到最后得到的最小均方误差小于某个阈值。此时，聚类中心逐渐稳定，可以结束聚类。

4.2 热点问题的发现

将附件三的文本留言进行数据预处理，转化为向量后，发现采用$K-Means$聚类算法将其分成六类时，聚类效果最好。得到的散点图如下



图十 聚类散点图

图中六个颜色代表着$K-Means$算法聚成的六类。接着根据热度热度评价指标公式:

$$y = \left[\left(\frac{a}{a_n} + \frac{b}{b_n} + \frac{n}{N}\right) \div 3\right] \times 1000 \quad (4\text{-}3)$$

其中：a, b, n表示此类所获点赞数，反对数，总点数，a_n, b_n, c_n代表所有留言文本总赞数，总反对数，以及总点数。算出它们的热度值，并进行排序得到排名前五的热度问题。见表二，详细见上传的“热点问题表.xls”。

表二 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	1	693.83683	2019/07/21 至 2019/09/25	A5 区劳动东路魅力之城小区	油烟扰民污染空气
2	2	568.8852155	2019/11/02 至 2020/01/02	A2 区丽发新城	搅拌厂噪音扰民灰尘污染
3	3	539.374325782093		A 市伊景园滨河苑	开发商违规销售涨价
4	4	518.4728317	2019/07/18 至 2019/07/23	A3 区西湖街道	拆迁停滞村民居无定所
5	5	462.320850670365	2019/01/09 至 2019/07/09	A7 县新国道 107	相关政府不同意拆迁

“热点问题留言明细表.xls”见上传文件，这里就不过多赘述。

五、答复意见的评价

评价答复意见需要从“相关性”,“完整性”,“可解释性”三个方面进行讨论，要给出评价方案就需要对这三个词有正确充分的理解。同时，要将三个性进行量化，从而给出评价方案。

5.1 相关性

在《现代汉语词典》中，对相关一词的解释是彼此关联。而在生活中，人们对相关性的认识通常是直觉性，习惯性的理解，由于频繁的使用，这个较生活化的词，难以定义，其实也无需定义。因为它就像“非常”,“可以”这一类词在生活中的定义就如同自然科学中的公理一样。

为了计算文本间的相关性，使之与相似性有区别，就需要更深入判别两个文本语义之间关系。本文采用向量空间模型，对附件四的留言详情与回复文本进行相关性计算，并且给出优劣之分。

5.1.1 基于相关性对答复意见的评分步骤

Step1:进行数据预处理，并且通过$TF-IDF$权值向量方法[4]将所有文本转化为

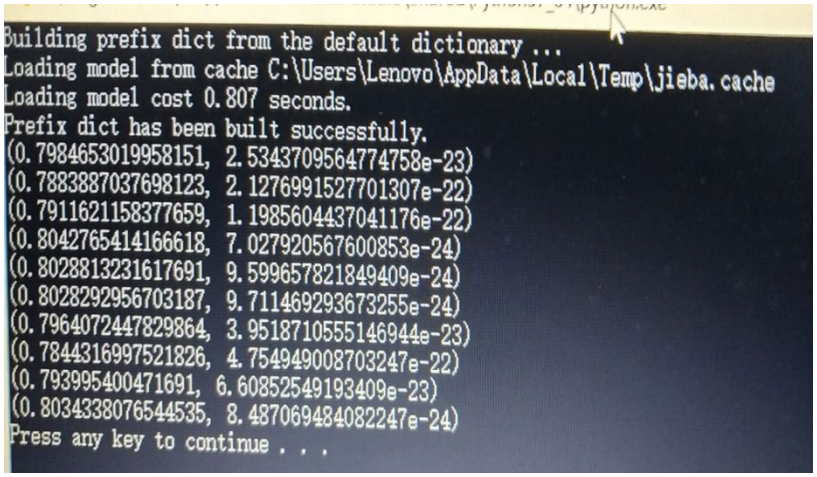
向量即两个文本的向量为$A_i = \{a_{1,i}, a_{2,i}, \dots, a_{n,i}\}$, $A_j = \{a_{1,j}, a_{2,j}, \dots, a_{n,j}\} n \leq m$。

Step2：采用向量空间模型进行计算，此处本文比较两个向量间的夹角偏差程度，使用的计算公式为

$$\rho(X,\ Y) = \frac{E[(X-u_X)(Y-u_Y)]}{\sqrt{\sum_{i=1}^{n}(X_i-u_X)^2}\sqrt{\sum_{i=1}^{n}(Y_i-u_Y)^2}} \quad (5\text{-}1)$$

通过 *python* 计算得到结果。

以附件四前十条留言为例，通过$Python$编程计算得到它们的相关性：



图十一 试例的相关性

Step3:将完整性进行区间划分，具体评分规则见下表：

表三 具体评分标准

相关性数值	低于0.5	(0.5,0.6]	(0.6,0.7]	(0.7,0.8]	(0.8,0.9]	(0.9,1.0]
所得分	0 分	1 分	2 分	3 分	4 分	5 分

答复留言由相关性得到的分值记为G_1。

5.2 完整性

政府问政平台的留言回复通常具有格式进行规范，而民众去问政平台的留言通常分为四类：咨询类留言，建言类留言，投诉类留言以及求助类留言。

每种留言都有相应的回答模板，根据人民网查询调查可得，具体见下表：

表四 相应留言回复模板

留言类别	回复模板
咨询类留言	*a*. 尊敬的问候；*b*. 依据政策或法规条文回答；*c*. 留下部门联系方式，以便留言者后续提问。
建言类留言	*a*. 尊敬问候；*b*. 对建言所针对的问题做出回应；*c*. 分析建言的现实操作性；*d*. 对建言表示感谢。
投诉类留言	*a* 尊敬问候；*b*. 对建言所针对的问题做出回应；*c*. 分析建言的现实操作性；*d*. 对建言表示感谢。
求助类留言	*a*. 尊敬问候；*b*. 表明调查主体部门；*c*. 可以直接解决的，迅速解决；需要个人参与解决的，则提供建议；*d*. 留下部门联系方式，以便留言者后续求助。

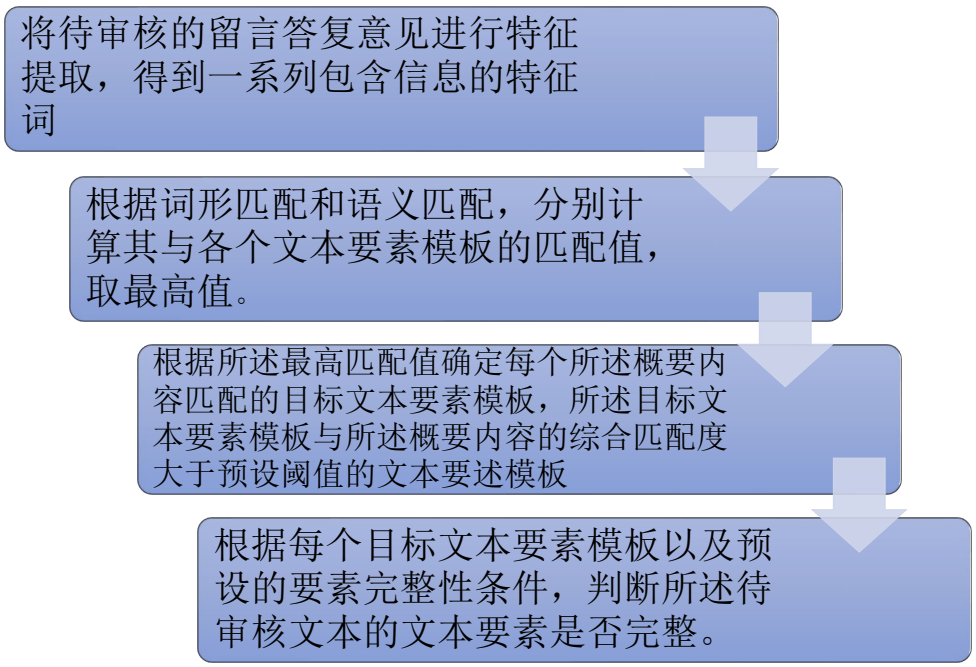
基于此，本文采用使用深度学习模型计算文本的完整性。

5.2.1 基于完整性对答复意见的评分步骤

Step1:首先对附件四的所有答复留言本文进行数据预处理。

Step2：采用深度学习算法计算每个回复留言文本的完整性。

具体操作如下图：



图十二 深度学习算法步骤

Step3：根据最高的匹配值按照表三，进行评分，记为G_1。

5.3 可解释性

回复留言的可解释性是指在回复民众留言时，对相关留言内容的解释。例如在咨询“买房是否能够进行人才购房补助”的回复中，留言引用了“*A* 市人才购房及购房补贴实施办法（试行）》第七条规定”，进行说明，这样的留言回复就有相当的可解释性。为了量化可解释性，本问采用了深度神经网络进行学习，并计算留言文本的可解释性。

5.3.1 基于可解释性对答复意见的评分步骤

*Step*1：将附件四留言进行数据预处理，提取特征值。

*Step*2： 构建可解释性留言文本的要素语料库，对于可解释性的计算，本文基于深度神经网络建立模型，并对留言答复评论进行可解释性的要素识别，计算出其可解释性。

*Step*3:通过表三进行评分，得到数值为G_3。

最后答复留言总得分$G = G_1 + G_2 + G_3$。总得分为 15 分，按照表五的评判指标对回复质量进行评价。

表五 答复意见质量评价

回复总得分G	9 分以下	9~12	12 分以上
相关评价	不及格	良好	优秀

六、参考文献

[1]宿绍勋. 基于 SVM 文本分类的传销识别研究[D].北京工业大学,2019.、

[2]陈杨楠. 基于改进 SVM 算法的投诉文本分类研究[D].合肥工业大学,2019.

[3]张满堂. 基于改进 K-Means 的新闻聚类算法研究[D].燕山大学,2019

[4]白振田. 基于向量空间模型与规则匹配相结合的文本层次分类系统的研究[D].南京农业大学,2006.

七、附录

问题一：

```
import pandas as pd
import jieba
import re


def data_process(file='F:/c 语 言 编 译 /PythonApplication1/PythonApplication1/附件 2.xlsx'):
    #导入文件
    data=pd.read_excel(file)

stopWords=pd.read_csv('stopword.txt',header=None,sep='hahahahah')
    col=['留言详情','一级标签']
    #将附件 2 的 6 个中文标题化为英文标题
    data=data[col]
    data.columns=['label5','label6']
    data['label6_id']=data['label6'].factorize()[0]
    #去除其中的各种符号
    data_quchu1=data['label5'].apply(lambda x:re.sub('[a-zA-Z0-9\t\n\r,。、:！？；《》（）. :/“”.]','',x))
    data_quchu=data_quchu1.apply(lambda x:str(x).replace('\u3000','').replace('\xa0',''))
    #利用 jieba 库分词
    data_fenci=data_quchu.apply(lambda x:jieba.lcut(x))
    #将分词后的结果进行去停用词
    stopWords=list(stopWords.iloc[:,0])+['张','邓','书记','局长','您好','厅长','你好']
    data_quci=data_fenci.apply(lambda x:[i for i in x if i not in stopWords])
    adata=data_quci.apply(lambda x:' '.join(x))
    label6_id_df = data[['label6','label6_id']].drop_duplicates().sort_values('label6_id')
    label6_to_id = dict(label6_id_df.values)
    id_to_category = dict(label6_id_df[['label6_id','label6']].values)
    labels=data.label6_id
    return adata,labels,data,label6_id_df,label6_to_id,id_to_category,data_quci
    #return data_quci


import numpy as np
import pandas as pd
```

```
from wentil import data_process
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer,TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_selection import chi2
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
import seaborn as sns
from sklearn import metrics
from gensim.models.word2vec import Word2Vec


#导入数据
adata,labels,data,label6_id_df,label6_to_id,id_to_category,data_quci=data_process()

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense,Embedding,LSTM,SpatialDropout1D,Bidirectional
from keras.utils.np_utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.layers import Dropout

MAX_NB_WORDS=60000
MAX_SEQUENCE_LENGTH=250
EMBEDDING_DIM = 2000


tokenizer = Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+,-./:;<=>?@[\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(adata)
word_index = tokenizer.word_index
X = tokenizer.texts_to_sequences(adata)
#填充 X,让 X 的各个列的长度统一
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH)
#多类标签的 onehot 展开
Y = pd.get_dummies(data['label6_id']).values
```

```
   X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =
0.1, random_state = 42)
   model = Sequential()
   model.add(Embedding(MAX_NB_WORDS,                    EMBEDDING_DIM,
input_length=X.shape[1]))
   model.add(SpatialDropout1D(0.2))
   lstm=LSTM(100, dropout=0.2, recurrent_dropout=0.2)
   model.add(Bidirectional(lstm))
   model.add(Dense(7, activation='softmax'))
   model.compile(loss='categorical_crossentropy',     optimizer='adam',
metrics=['accuracy'])
   epochs = 4
   batch_size = 128


   model.fit(X_train,              Y_train,              epochs=epochs,
batch_size=batch_size,validation_split=0.1,
                       callbacks=[EarlyStopping(monitor='val_loss',
patience=3, min_delta=0.0001)])


   y_pred = model.predict(X_test)
   y_pred = y_pred.argmax(axis = 1)
   Y_test = Y_test.argmax(axis = 1)
   from sklearn.metrics import accuracy_score, confusion_matrix
   #生成混淆矩阵
   conf_mat = confusion_matrix(Y_test, y_pred)
   fig, ax = plt.subplots(figsize=(10,8))
   sns.heatmap(conf_mat, annot=True, fmt='d',
               xticklabels=label6_id_df.label6_id.values,
yticklabels=label6_id_df.label6_id.values)
   plt.ylabel('实际结果')
   plt.xlabel('预测结果')
   plt.show()
   from  sklearn.metrics import classification_report
   print(classification_report(Y_test,
y_pred,target_names=label6_id_df['label6'].values))



   ,,,
   #转化为 td-idf 权值向量
```

```
   tfidf                                                                    =
TfidfVectorizer(sublinear_tf=True,min_df=3,norm='l2',ngram_range=(1,2))
   features=tfidf.fit_transform(adata).toarray()

   #选择模型为支持向量机
   model = LinearSVC()
   X_train,  X_test,  y_train,  y_test,  indices_train,  indices_test=
train_test_split(features,    labels,    data.index,    test_size=0.21,
random_state=0)
   model.fit(X_train, y_train)
   y_pred = model.predict(X_test)
   from sklearn.metrics import confusion_matrix
   conf_mat = confusion_matrix(y_test, y_pred)
   fig, ax = plt.subplots(figsize=(10,10))
   sns.heatmap(conf_mat,                                          annot=True,
fmt='d',xticklabels=label6_id_df.label6_id.values,
yticklabels=label6_id_df.label6_id.values)
   plt.ylabel('Actual')
   plt.xlabel('Predicted')
   plt.show()
   print(metrics.classification_report(y_test,                    y_pred,
target_names=data['label6'].unique()))
   '''

   '''
   for predicted in label6_id_df.label6_id:
    for actual in label6_id_df.label6_id:
      if predicted != actual and conf_mat[actual, predicted] >= 6:
        print("'{}'       predicted       as       '{}'       :       {}
examples.".format(id_to_category[actual],    id_to_category[predicted],
conf_mat[actual, predicted]))
        display(data.loc[indices_test[(y_test == actual) & (y_pred ==
predicted)]][['label6','label5']])
        print('')
   '''




   '''
   #朴素贝叶斯模型
   X_train,X_test,y_train,y_test=train_test_split(adata8,data['label6'
],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)
'''

'''
#测试 4 种不同的模型
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'])
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()
print(cv_df.groupby('model_name').accuracy.mean())
'''

'''
#训练词向量
corpus=[]
for i in range(len(adata)):
    corpus.append(adata[i])
sentences=[]
for i in range(len(data_quci)):
  sentences.append(data_quci[i])
cx=Word2Vec(sentences,size=100,sg=1,min_count=1,windows=5)
```

```
yy=[]
for i in cx.wv.vocab:
    yy.append(i)
vectorizer = CountVectorizer()
u = vectorizer.fit_transform(corpus)
word=vectorizer.get_feature_names()
transformer = TfidfTransformer()
tfidf = transformer.fit_transform(u)
print(u)
```

问题二：

```
import pandas as pd
import numpy as np
import jieba
import re
from gensim.models.word2vec import Word2Vec
import gensim

def data_process4(file='C:/Users/DELL1/Desktop/2020泰迪杯数据挖掘大赛/C题赛题/示例数据/附件3.xlsx'):
    #导入文件
    data=pd.read_excel(file)

stopWords=pd.read_csv('stopword.txt',header=None,sep='hahahahah',engine='python')
    #将附件2的6个中文标题化为英文标题

data.columns=['label1','label2','label3','label4','label5','label6','label7']
    #去除其中的各种符号
    data_quchu2=data['label5'].drop_duplicates()
    data_quchu1=data_quchu2.apply(lambda x:re.sub('[a-zA-Z0-9\t\n,。、:！？；《》（）. :/“”]','',x))
    data_quchu=data_quchu1.apply(lambda x:str(x).replace('\u3000','').replace('\xa0',''))
    #利用jieba库分词
    data_fenci=data_quchu.apply(lambda x:jieba.lcut(x))
    #将分词后的结果进行去停用词
    stopWords=list(stopWords.iloc[:,0])
    data_quci=data_fenci.apply(lambda x:[i for i in x if i not in stopWords])
    adata0=data_quci.apply(lambda x:' '.join(x))
```

```
    return adata0,data,data_quci

'''
#利用 Word2vec 训练词向量
adata,data,data_quci=data_process()
sentences=[]
#将 data_quci 的每个对象中的词语转换为一个单独的小列表
def list_of_groups(b, per_list_len): #per_list_len:  每个小列表的长
度
    list_of_group = zip(*(iter(b),) *per_list_len)
    end_list = [list(i) for i in list_of_group] # i is a tuple
    count = len(b) % per_list_len
    end_list.append(b[-count:]) if count !=0 else end_list
    return end_list
#将 data_quci 的所有对象中的词语都化为一个个小列表
b=data_quci.apply(lambda x:list_of_groups(x,1))
#将每个词语组成的小列表合并为对象 sentences 需要的语料
for j in range(30):
   for i in b[j]:
       sentences.append(i)
#用 word2vec 构建每个词语的词向量
model=Word2Vec(sentences,size=300,min_count=5,sg=0)
'''

'''
#计算相似度
def vector_similarity(s1, s2):
    model=moxing()
    def sentence_vector(s):
        v = np.zeros(100)
        for word in s:
          v += model[word]
        v /= len(s)
        return v
    v1, v2 = sentence_vector(s1), sentence_vector(s2)
    return np.dot(v1, v2) / (norm(v1) * norm(v2))

#a=pd.DataFrame(data_process())
#a.to_csv('F:/c          语          言          编          译
/PythonApplication1/PythonApplication1/fujian3.csv')
'''

from wenti22 import data_process4
```

```
import pandas as pd
import jieba
import numpy as np
import re
from gensim.models.word2vec import Word2Vec
#from scipy.linalg import norm
from sklearn.cluster import KMeans,MiniBatchKMeans
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer,TfidfVectorizer,HashingVectorizer
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
import jieba.analyse

#
adata0,data,data_quci=data_process4()
adata1=adata0.apply(lambda x:jieba.analyse.extract_tags(x))
adata=adata1.apply(lambda x:' '.join(x))

def redu():#基于点赞数以及反对数的热度指数的计算
    zhuti=quci()
    m=julei()
    id=Id()
    Hot=[]
    N=len(zhuti)
    an=sum(data['label6'])
    bn=sum(data['label7'])
    for i in id:
        a=sum(data['label6'].values[m[i]])
        b=sum(data['label7'].values[m[i]])
        n=len(m[i])
        if bn==0:
            Hot.append(int(1000*((a/an+n/N)/2)))
        else:
            Hot.append(int(1000*((a/an+b/bn+n/N)/3)))
    return Hot

def quci():#对所有数据进行去词处理

stopWords=pd.read_csv('stopword.txt',header=None,sep='hahahahah')
    jieba.load_userdict('Newword.txt')
    zhuti1=data['label3'].apply(lambda x:re.sub(', ','',x))
    zhuti2=zhuti1.apply(lambda x:jieba.lcut(x))
    stopWords=list(stopWords.iloc[:,0])
```

```
        zhuti=zhuti2.apply(lambda x:[i for i in x if i not in stopWords])
        return zhuti


    def julei(): #对所分类后的数据聚类
        tfidf                                                              =
TfidfVectorizer(sublinear_tf=True,min_df=3,norm='l2',ngram_range=(1,2))
        weight=tfidf.fit_transform(adata).toarray()
        tsne = TSNE(n_components=2)
        decomposition_data = tsne.fit_transform(weight)
        # kmeans 聚类
        # print data
        kmeans                  =                  KMeans(n_clusters=2000,
random_state=2000).fit(decomposition_data)#k 值可以自己设置，不一定是五
类
        # print kmeans
        centroid_list = kmeans.cluster_centers_
        labels = kmeans.labels_
        n_clusters_ = len(centroid_list)
        #print ("cluster centroids:",centroid_list)
        #print(labels)
        max_centroid = 0
        max_cluster_id = 0
        cluster_menmbers_list = []
        for i in range(0, n_clusters_):
         menmbers_list = []
         for j in range(0, len(labels)):
          if labels[j] == i:
           menmbers_list.append(j)
         cluster_menmbers_list.append(menmbers_list)
        m=cluster_menmbers_list
        #聚类结果

        tsne = TSNE(n_components=2)
        decomposition_data = tsne.fit_transform(weight)

        x = []
        y = []

        for i in decomposition_data:
            x.append(i[0])
            y.append(i[1])
```

```
        fig = plt.figure(figsize=(10, 10))
        ax = plt.axes()
        plt.scatter(x, y, c=kmeans.labels_, marker="x")
        plt.xticks(())
        plt.yticks(())
        plt.show()

        return m
    m = julei()

    def Id():#问题 ID
        m=julei()
        id=[]
        for i in range(len(m)):
          a=id.append(i)
        return id

    def Data():#根据附件一标签提取数据
        m=julei()
        a=data.values[m[0]]
        return a

    def transform(adata,n_features=1000):#将文档转化为向量
        vectorizer = TfidfVectorizer(max_df=0.5, max_features=n_features,
min_df=2,use_idf=True)
        X = vectorizer.fit_transform(adata)
        return X,vectorizer

    def  train(X,vectorizer,true_k=10,minibatch  =  False,showLable  =
False):#使用采样数据还是原始数据训练 k-means,
        if minibatch:
            km  =  MiniBatchKMeans(n_clusters=true_k,  init='k-means++',
n_init=1,
                                   init_size=1000,        batch_size=1000,
verbose=False)
        else:
            km     =     KMeans(n_clusters=true_k,     init='k-means++',
max_iter=300, n_init=1,
                          verbose=False)
        km.fit(X)
        if showLable:
            print("Top terms per cluster:")
            order_centroids = km.cluster_centers_.argsort()[:, ::-1]
```

```
        terms = vectorizer.get_feature_names()
        print (vectorizer.get_stop_words())
        for i in range(true_k):
            print("Cluster %d:" % i, end='')
            for ind in order_centroids[i, :15]:
                print(' %s' % terms[ind], end='')
            print()
    result = list(km.predict(X))
    print ('Cluster distribution:')
    print (dict([(i, result.count(i)) for i in result]))
    return -km.score(X)

def test():#测试选择最优参数
    '''测试选择最优参数'''
    print("%d documents" % len(adata))
    X,vectorizer = transform(adata,n_features=500)
    true_ks = []
    scores = []
    for i in range(3,80,1):
        score = train(X,vectorizer,true_k=i)/len(adata)
        print (i,score)
        true_ks.append(i)
        scores.append(score)
    plt.figure(figsize=(8,4))
    plt.plot(true_ks,scores,label="error",color="red",linewidth=1)
    plt.xlabel("n_features")
    plt.ylabel("error")
    plt.legend()
    plt.show()

def out():#在最优参数下输出聚类结果
    '''在最优参数下输出聚类结果'''
    X,vectorizer = transform(adata,n_features=500)
    score = train(X,vectorizer,true_k=4000,showLable=True)/len(adata)
    print (score)

print(julei())
print(out())

#
# for i in range(6):
#     for j in range(len(m[i])):
#         m[i][j]
```

```
#           print(m[i][j])

问题三：
import pandas as pd
import numpy as np
import jieba
import re



def       data_process1(file='F:/c       语       言       编       译
/PythonApplication1/PythonApplication1/附件 4.xlsx'):
      #导入文件
      data=pd.read_excel(file)

stopWords=pd.read_csv('stopword.txt',header=None,sep='hahahahah',engin
e='python')
      #将附件 2 的 6 个中文标题化为英文标题

data.columns=['label1','label2','label3','label4','label5','label6','l
abel7']
      #去除其中的各种符号
      data_quchu1=data['label5'].apply(lambda      x:re.sub('[a-zA-Z0-
9\t\n,。、:！？；《》（）. :/“”%*]','',x))
      data_quchu=data_quchu1.apply(lambda
x:str(x).replace('\u3000','').replace('\xa0','').replace(' ',''))
      #利用 jieba 库分词
      data_fenci=data_quchu.apply(lambda x:jieba.lcut(x))
      #将分词后的结果进行去停用词
      stopWords=list(stopWords.iloc[:,0])
      data_quci1=data_fenci.apply(lambda x:[i for i in x if i not in
stopWords])
      adata1=data_quci1.apply(lambda x:' '.join(x))
      return adata1,data,data_quci1

def       data_process2(file='F:/c       语       言       编       译
/PythonApplication1/PythonApplication1/附件 4.xlsx'):
      #导入文件
      data=pd.read_excel(file)

stopWords=pd.read_csv('stopword.txt',header=None,sep='hahahahah',engin
e='python')
      #将附件 2 的 6 个中文标题化为英文标题
```

```
data.columns=['label1','label2','label3','label4','label5','label6','label7']
        #去除其中的各种符号
        data_quchu1=data['label6'].apply(lambda x:re.sub('[a-zA-Z0-9\t\n,。、:！？；《》（）. :/“”]','',x))
        data_quchu=data_quchu1.apply(lambda x:str(x).replace('\u3000','').replace('\xa0',''))
        #利用 jieba 库分词
        data_fenci=data_quchu.apply(lambda x:jieba.lcut(x))
        #将分词后的结果进行去停用词
        stopWords=list(stopWords.iloc[:,0])
        data_quci2=data_fenci.apply(lambda x:[i for i in x if i not in stopWords])
        adata2=data_quci2.apply(lambda x:' '.join(x))
        return adata2,data,data_quci2

    def data_process3(file='F:/c 语 言 编 译/PythonApplication1/PythonApplication1/附件 4.xlsx'):
        #导入文件
        data=pd.read_excel(file)

stopWords=pd.read_csv('stopword.txt',header=None,sep='hahahahah',engine='python')
        #将附件 2 的 6 个中文标题化为英文标题

data.columns=['label1','label2','label3','label4','label5','label6','label7']
        #去除其中的各种符号
        data_quchu2=data['label3'].drop_duplicates()
        data_quchu1=data_quchu2.apply(lambda x:re.sub('[a-zA-Z0-9\t\n,。、:！？；《》（）. :/“”]','',x))
        data_quchu=data_quchu1.apply(lambda x:str(x).replace('\u3000','').replace('\xa0',''))
        #利用 jieba 库分词
        data_fenci=data_quchu.apply(lambda x:jieba.lcut(x))
        #将分词后的结果进行去停用词
        stopWords=list(stopWords.iloc[:,0])
        data_quci3=data_fenci.apply(lambda x:[i for i in x if i not in stopWords])
        adata3=data_quci3.apply(lambda x:' '.join(x))
        return data_quci3,adata3
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import
CountVectorizer,TfidfTransformer,TfidfVectorizer
from wenti3 import data_process1,data_process2,data_process3
from gensim.models.word2vec import Word2Vec
from scipy.stats import pearsonr
import re
import jieba.analyse

adata1,data,data_quci1=data_process1()
adata2,data,data_quci2=data_process2()
adata3,data_quci3=data_process3()
#训练词向量
sentences=[]
corpus=[]
for i in range(2816):
    sentences.append(data_quci1[i])
for j in range(2816):
    corpus.append(data_quci2[j])
model1=Word2Vec(sentences,size=100,window=5,min_count=1)
model2=Word2Vec(corpus,size=100,window=5,min_count=1)
#评价相关性
def xg(h):
    z=jieba.analyse.extract_tags(adata1[0],withWeight=True)
    zz=jieba.analyse.extract_tags(adata2[0],withWeight=True)
    vectorizer =
TfidfVectorizer(min_df=1,use_idf=True,max_features=100)
    #aaa=vectorizer.fit_transform(adata1).toarray()
    #bbb=vectorizer.fit_transform(adata2).toarray()
    a=np.zeros(100)
    for j in z:
        if j[0] in model1.wv.vocab:
            aa=j[1]*model1.wv[j[0]]
            a+=aa
    b=np.zeros(100)
    for k in zz:
        if k[0] in model2.wv.vocab:
            bb=k[1]*model2.wv[k[0]]
            b+=bb
    #q1=a+aaa[h]
    #q2=b+bbb[h]
```

```
        m=pearsonr(a,b)
        return m[0]

    #相关性的可视化
    def p():
        a1=0
        a2=0
        a3=0
        a4=0
        a5=0
        for i in range(50):
            if xg(i)<0.5:
                a1+=1
            if(xg(i)>0.5 and xg(i)<0.6):
                a2+=1
            if(xg(i)>0.6 and xg(i)<0.7):
                a3+=1
            if(xg(i)>0.7 and xg(i)<0.8):
                a4+=1
            if(xg(i)>0.9 and xg(i)<1.0):
                a5+=1
        x=[1,2,3,4,5]
        y=[a1,a2,a3,a4,a5]
        plt.bar(x,y)
        plt.show()
    #完整性
    #提取数据特征
    tfidf                                                                    =
TfidfVectorizer(sublinear_tf=True,min_df=3,norm='l2',ngram_range=(1,2))
    features=tfidf.fit_transform(adata2).toarray()



    #可解释性
    from xgboost import XGBClassifier
    from sklearn.model_selection import train_test_split
    import eli5 # python 计算 permutation importance 工具包
    from eli5.sklearn import PermutationImportance

    path = 'C:/Users/DELL1/Desktop/2020 泰迪杯数据挖掘大赛/C 题赛题/示例
数据/附件 4.xlsx'
    data = pd.read_csv(path)
    #...省略数据预处理过程
    X_train,  X_test,  y_train,  y_test  =  train_test_split(df,  y,
```

```
test_size=0.2, random_state = 400)

   # 训练 XGBoost 模型
   model = xgb.XGBClassifier(
       learning_rate =0.05,
       n_estimators=100,
       max_depth=3,
       min_child_weight=1,
       gamma=0.3,
       subsample=0.8,
       colsample_bytree=0.8,
       objective= 'multi:softprob',
       nthread=4, scale_pos_weight=1,
       num_class=2,
       seed=27
       ).fit(X_train, y_train)
   perm = PermutationImportance(model, random_state = 1).fit(X_test,
y_test) # 实例化
   eli5.show_weights(perm, feature_names = X_test.columns.tolist())


   '''
   def fg(i):
       pattern     =     r',|/|    ;     |\'|`|\[|\]|<|>|    ?     |    :
|″|\{|\}|\~|!|@|#|\$|%|\^|&|\(|\)|-|=|\_|\+|，|。|、|；| ‘|’ |【|】|·|！
| |…| （|）'
       result_list1 = re.split(pattern,data['label6'][i])
       data_quchu1=data['label5'].apply(lambda x:re.sub('[\t\n]','',x))
       data_quchu=data_quchu1.apply(lambda
x:str(x).replace('\u3000','').replace('\xa0','').replace(' ',''))
       result_list2 = re.split(pattern,data_quchu[i])
       return result_list1,result_list2
   '''
```