基于文本挖掘的智慧政务留言处理模型

摘要

本文基于文本挖掘，构建智慧政务留言处理模型，智慧分类留言信息、整理民意热点，同时评价政府答复意见，以提高政府的施政效率和管理水平。

针对问题一，我们将给定的三级标签体系作为基础分类关键词，在留言中进行匹配，从词频和词关联度两个方面分别构建 TF-IDF 权重表、词频权重表，和 TextRank 权重表。首先，借助最近邻算法、随机森林算法、梯度提升算法和支持向量机算法分别进行拟合分类，以最近邻算法的分类结果作为一级标签时，其总的 F-Score 的值为 0.8729。其次我们将留言分为五类：留言群 1 占比 13%，F-score 为 0.5143；留言群 2 占比 4%，F-score 为 0.7135；留言群 3 占比 4%，F-score 为 0.5324；留言群 4 占比 1%，F-score 为 0.8448；留言群 5 占比 78%，F-score 为 0.9499。同时我们对三类权重表的构建进行了灵敏性分析，数值例子表明该分类模型是有效的。最后，选取每类留言群中使 F-score 值最高的分类模型，将所有模型重组，得到总的 F-Score 值为 0.8904。

针对问题二，构建头部热点问题筛选模型，从事发地点和时间入手，先地区，后小区进一步构造特征，再利用保留高频词的方法对热点问题进行捕捉。以类作为单位，综合考虑回复时间跨度，点赞反对数以及某个热点问题的样本容量三个维度，利用熵值法+优劣解距离法（TOPSIS）方法进行评价，最终确定五类热点问题。

针对问题三，从相关性、完整性、可解释性以及时效性四个角度评价答复意见：1）从留言主题和留言详情入手构造相关性指标，通过 TF-IDF 模型从相关词的词频权重分别考察两者与答复意见间的相关度；2）从答复意见的结构完整性考虑完整性的评价，以尊称、留言主题、回复内容的开头、感谢语和时间建立 5 个指标；3）可解释性强的答复意见主要是由较多的短句连接形成的具有逻辑性的长叙述，因此该指标使用标点符号刻画文本结构并考量有效答复文本与留言文本长度比；4）“时效性”指标直接衡量答复的及时程度。将四类指标得分规范至 0-1 区间后求和，按照总分的大小将各条答复意见分为四级，3 分以上为“完美”，其余一分一级由高到低为“较好”、“达标”和“待改善”。

关键词：TF-IDF、TextRank、最近邻、熵值法、TOPSIS、答复评价模型

目录

1. 引言

1.1. 问题背景

近年来，随着社会的发展，政府越来越注重对民意的了解，通过微博、微信、阳光热线、市长信箱等多个网络问政平台汇聚民智，鼓励群众问政。目前，相关的社情民意反映主要依靠人工进行留言划分和热点整理，网络问政平台的开放使得各类相关文本数据量大幅增加，这给政府的相关工作部门带来了极大的挑战。与此同时，人工智能不断发展，人脸识别、智能翻译等人工智能应用极大程度上便利了人们的生活，将人工智能与政务系统相结合成为了政府管理的新方向。基于自然语言处理技术的智慧政务系统，采用自然语言处理和文本挖掘的方法处理文本数据，将有效地缓解目前相关工作部门的人工整理压力，推动政府管理水平的提升，提高政府的施政效率。

1.2. 问题重述

问题一：目前，政府工作人员在处理网络问政平台的群众留言时，首先按照三级划分体系对留言进行分类，再将留言分派至相应的职能部门处理。这样的留言分类处理方式存在工作量大、效率低、差错率高等问题。因此，需要根据已有的留言数据建立留言分类模型，并使用 F-Score 评价分类模型。

问题二：在某一时段内，网络问政平台上群众集中反映的某一问题叫做热点问题，及时发现热点问题集中分派至相应的职能部门进行解决将有助于提升政府的服务效率。因此，需要根据已有的留言数据，归纳某一时段内反映特定人群或地点问题的留言，同时定义合理的热度评价指标，并给出热点问题的评价结果。

问题三：尝试设计对相关部门留言答复意见的评价方案，评价方案需要考虑答复的相关性、完整性、可解释性等角度，并依据已有的答复意见数据实现评价方案。

2. 模型说明

2.1. 符号说明

P_i	第 i 类的查准率
R_i	第 i 类的查全率
TF	词频

IDF	逆向文件频率
|D|	语料库中的文件总数
Out(Vi)	点 Vi 指向的点集合
d	阻尼系数

2.2. 模型假设

- ✧ 假设提供的附件 2 中留言的一级分类标签无误；
- ✧ 假设同类留言存在一定的相似性，具有同类高频关键词；
- ✧ 假设用户留言时会详细阐述事发地点；
- ✧ 假设构建的指标能够全面的对答复意见的质量展开评价。

3. 问题一求解

3.1. 问题分析

对于留言分类问题，为实现自动划分，需先将文本转换为向量形式，故根据提供的 15 个一级分类标签构建 15 个指标。依据已提供的三级标签体系，对留言主题与留言详情进行匹配，从而构建基础权重表，再根据留言内容采用 TF-IDF 算法提取关键词，并利用关键词进行匹配并实现对基础权重表的二次填充，得到 TF-IDF 权重表。此外，根据最高词频以及 TextRank 算法提取关键词，分别利用关键词进行匹配，从而构建词频权重表和 TextRank 权重表。依据三类权重表，分别调用算法进行拟合分类，寻找最优算法。接着对留言进行划分，即为保证分类的准确性，减轻工作量，提高效率，在分类中使某类留言群具有高分类准确性，即可以根据程序实现分类，其他留言群则需人工处理。最后展开灵敏性分析，并基于上述分析，综合之前所有的分类模型分别对留言群进行分类，选取每类留言群中使 F-score 值最高的分类模型进行组合，从而构建最终的分类模型。

3.2. 权重表构建

3.2.1. 构建基础权重表

根据附件 1 的 15 个一级分类标签名，构建出 15 个指标，即每条留言对应 15 个指标，初值为 0，从而形成一张权重表，如表 1 所示：

表 1 权重表

城乡建设	党务政务	国土资源	环境保护	纪检监察	交通运输	经济管理	科技与信息产业	民政	农村农业	商贸旅游	卫生计生	政法	教育文体	劳动和社会保障
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

根据附件 1 的分类三级标签体系，对留言的主题和详情进行文字匹配，并展开权重值的初步填充。例如对于第一条留言，先根据附件 1 中第一行的一级分类，即“城乡建设”进行匹配，若该标签名在留言中出现，则将权重表第一行中对应标签名的权重值+a，接着根据第一行的二级分类，即“安全生产”进行匹配，若该标签名在留言中出现，将权重表第一行中对应标签名的权重值+b，再根据第一行的三级分类，即“事故处理”进行匹配，若该标签名在留言中出现，将权重表第一行中对应标签名的权重值+c。在匹配的过程中，如果匹配词多次出现，则依据对应的权重值重复累加。根据附件 1 中每一行的标签名分类依次在文本中进行匹配，计算对应的权重值相加并填充至指定的位置。为提高处理效率，在匹配的过程中若该行的一级分类标签名与上一行的一级分类标签名相同则跳过，不再进行匹配，相应的二级分类标签名与三级分类标签名也采用同样的方法处理。

基于划分体系，假设各级标签在留言中的出现对留言是否分为该类别存在不同的影响，即如果留言中含有“城乡建设”这四个字，可能表明这条留言大概率属于“城乡建设”这一类。一般来说，一级标签的重要性>二级标签的重要性>三级标签的重要性，故先假设权重值 a 为 10，b 为 5，c 为 1，后期将对权重值进行修改，展开灵敏性分析，从而检验权重值的可靠性。

此外，通过对所提供的标签名的分析，可以发现城乡建设与经济管理这两大一级分类指标下存有相同的二级分类标签名和三级分类标签名，故对于留言进行筛选，若留言主题和留言详情中不包含“经济”或“元”一词，则将经济管理指标对应的权重值重设为 0，得到初步的权重表如表 2 所示：

表 2 基础权重表

城乡建设	党务政务	国土资源	环境保护	纪检监察	交通运输	经济管理	科技与信息产业	民政	农村农业	商贸旅游	卫生计生	政法	教育文体	劳动和社会保障
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

根据基础权重表，对附件 2 中的所有留言进行分类，即给 9210 条留言添加标签。在分类过程中，取每条留言的最大权重值对应的指标名作为该条留言的标签名，并将分类标签与实际标签进行比较，通过如下公式计算 F-Score 的值评价基础权重表分类的效果：

$$F = \frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i+R_i} \quad (1)$$

其中 P_i 为第 i 类的查准率, R_i 为第 i 类的查全率。通过计算得到 F-Score 的值为 0.2902，得分并不高。通过回查基础权重表，可以发现某些留言存在所有指标值均为 0 的情况，即留言主题和留言详情中不包含各级标签，为得到有效的基础权重表分类效果，故将这部分留言删除，将剩余的 3345 条留言进行分类，重新计算得到 F-Score 的值为 0.5926。由此可见，基础权重表对留言的分类会产生一定影响，故接下来需要在基础权重表的基础上进行填充完善，从而提高留言分类的准确度，获得更高的 F-Score 值。

3.2.2. 构建 TF-IDF 权重表

在基础权重表的构建中只考虑了各级分类标签名在留言主题和留言详情中的频率，因此为提高分类准确率，接下来利用 python 对留言文本进一步处理。首先依照附件 2 中的一级标签每一类的留言文本分别进行正则处理，再利用 jieba 分词包提取文本中的名词和动词，并采用 TF-IDF 算法对提取出的词语计算对应的 TF-IDF 值。

TF-IDF（term frequency－inverse document frequency，词频-逆向文件频率）是一种用于信息检索与文本挖掘的常用加权技术。该算法用以评估一个字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反

比下降。该算法的公式为：

$$TF-IDF=tf_{i,j}\times idf_i=\frac{n_{i,j}}{\sum_k n_{k,j}}\times\log\frac{|D|}{|\{j:t_i\in d_j\}|} \quad (2)$$

利用 python 提取 TF-IDF 值最大的 500 个关键词，即每一类留言都得到 500 个关键词，并按照各类留言关键词的 TF-IDF 值由大到小的顺序分别对每一类留言的 500 个关键词进行赋权，通过多次试验对比，最终选择设权重值为 1~$\frac{2}{7}$的一串等差数值。在后期进行灵敏性分析中，也可以验证该权重值取值的可靠性。

基于基础权重表，根据提取的关键词对留言主题和留言详情进行再匹配，展开各留言权重的二次填充。如对于第一条留言，若含有“城乡建设”这类标签名所对应的 500 个关键词中的某几个关键词，则权重表第一行中“城乡建设”指标所对应的权重值加上相应关键词的权重值，如果存在重复出现的关键词，则权重值多次累加。此外，若该条留言同时含有其他类标签名对应的一些关键词，则权重表第一行中对应类标签名下的权重值也需加上相应关键词的权重值。最终得到 TF-IDF 权重表如表 3 所示：

表 3 TF-IDF **权重表**

城乡建设	党务政务	国土资源	环境保护	纪检监察	交通运输	经济管理	科技与信息产业	民政	农村农业	商贸旅游	卫生计生	政法	教育文体	劳动和社会保障
13.35	0	0	7.572857	0	9.28	0	0	0	0	3.884286	1.294286	0	2.32	2.827142857
11.09857	0	0	8.455714	0	8.482857	0	0	0	0	6.794286	4.262857	0	4.574286	3.528571429
212.5857	0	0	125.3186	0	123.7357	0	0	0	0	193.41	95.05	0	130.3057	93.05571429
13.31857	0	0	14.23	0	10.79143	0	0	0	0	13.57286	9.484286	0	11.78143	9.904285714
12.89429	0	0	13.81429	0	10.43	0	0	0	0	13.57286	9.484286	0	11.78143	9.904285714
35.02143	0	0	20.56429	0	19.93	0	0	0	0	33.36714	12.45143	0	18.65714	15.79571429
8.021429	0	0	4.457143	0	5.618571	0	0	0	0	4.905714	3.601429	0	5.392857	3.737142857
31.50143	0	0	21.99	0	20.95	0	0	0	0	25.48	16.43	0	21.9	20.017142857
94.45714	0	0	79.91714	0	61.48571	0	0	0	0	76.47857	40.61143	0	59.22429	42.18571429
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

根据 TF-IDF 权重表对附件 2 中的所有留言进行分类，取每条留言的最大权重值对应的指标名作为该条留言的标签名，并记分类结果为“TF-IDF 分类”，最终计算得到 F-Score 的值为 0.8512。

3.2.3. 构建词频权重表

考虑到各类留言可能存在一些共同的高词频词汇，如“我们”、“市”等，

但对留言的分类并没有实际的作用，因此先根据附件 2 中的留言，利用 jieba 分词包提取词频最高的 15 个词，将这 15 个词作为去除词，即去除各类留言中基本含有的这些词，再根据附件 2 中留言的一级分类标签名，将留言进行分类。使用 python 将每一类的留言文本分别进行正则处理，并利用 jieba 分词包进行词的划分，将划分后的词进一步处理，即删去停词，连接词以及上述构建的去除词等等。接着统计每一类中词频最高的 500 个词，以此作为关键词，并按照关键词词频由多到少的顺序对关键词进行赋权，通过多次试验对比，最终选择设权重值为 1~$\frac{2}{7}$ 的一串等差数值。

基于基础权重表，根据提取的关键词对留言主题和留言详情进行匹配，展开权重的二次填充。得到词频权重表如表 4 所示：

表 4 词频权重表

城乡建设	党务政务	国土资源	环境保护	纪检监察	交通运输	经济管理	科技与信息产业	民政	农村农业	商贸旅游	卫生计生	政法	教育文体	劳动和社会保障
22.80857	0	0	12.42571	0	14.91286	0	0	0	0	11.03143	3.808571	0	8.271429	6.228571429
16.67429	0	0	14.16571	0	11.56857	0	0	0	0	10.48	5.722857	0	7.892857	5.347142857
239.2143	0	0	132.2471	0	173.9271	0	0	0	0	228.3314	112.7729	0	134.1586	120.9857143
18.1	0	0	20.92714	0	9.33	0	0	0	0	13.32714	8.885714	0	9.931429	6.038571429
18.43857	0	0	20.92714	0	9.33	0	0	0	0	12.69286	8.885714	0	9.931429	6.038571429
38.47143	0	0	24.37857	0	24.32714	0	0	0	0	34.56	14.98857	0	18.31857	17.44
8.127148	0	0	4.332857	0	5.122857	0	0	0	0	5.807143	4.428571	0	4.675714	4.977142857
35.51714	0	0	30.28714	0	23.80143	0	0	0	0	25.84143	17.46429	0	20.24571	22.27285714
97.35	0	0	79.31286	0	69.88714	0	0	0	0	84.14714	48.31	0	65.72429	48.79142857
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

根据词频权重表对附件 2 中的所有留言进行分类，取每条留言的最大权重值对应的指标名作为该条留言的标签名，并记分类结果为“词频分类”，最终计算得到 F-Score 的值为 0.8283。

3.2.4. 构建 TextRank 权重表

由于中文表达的特殊性，有时候词频无法完全刻画关键词，此处引入 TextRank 算法构建权重表，考虑与别的词关联性越大的词越重要。TextRank 算法是一种用于文本的基于图的排序算法。其基本思想来源于谷歌的 PageRank 算法，通过把文本分割成若干组成单元(单词、句子)并建立图模型，利用投票机制对文本中的重要成分进行排序。该算法的公式为：

$$WS(V_i) = (1-d) + d \times \frac{w_{ji}}{\sum_{V_i \in Out(V_j)} w_{jk}} WS(V_j) \quad (3)$$

与 TF-IDF 权重表构建流程一致，在构建 TextRank 权重表时，首先利用 python 对每一类提取出的词语计算对应的 TextRank 值，并提取 TextRank 值最大的 500 个关键词，即每一类留言都得到 500 个关键词。基于基础权重表，根据提取得到的关键词对留言主题和留言详情进行匹配，实现对各留言权重的二次填充，最终得到 TextRank 权重表如表 5 所示：

表 5 TextRank 权重表

城乡建设	党务政务	国土资源	环境保护	纪检监察	交通运输	经济管理	科技与信息产业	民政	农村农业	商贸旅游	卫生计生	政法	教育文体	劳动和社会保障
13.83286	0	0	8.128571	0	7.178571	0	0	0	0	6.121429	1.898571	0	4.624286	4.522857143
11.63571	0	0	9.575714	0	8.72	0	0	0	0	8.052857	4.517143	0	5.361429	4.185714286
212.8229	0	0	136.2743	0	138.88	0	0	0	0	191.0614	104.7057	0	139.9029	103.1085714
13.15857	0	0	14.80143	0	12.65571	0	0	0	0	14.5	10.54143	0	12.90571	10.86857143
13.15857	0	0	14.46286	0	12.65571	0	0	0	0	14.5	10.54143	0	12.90571	10.86857143
33.83429	0	0	21.07857	0	21.11429	0	0	0	0	32.46286	14.39714	0	20.32714	16.41
9.241429	0	0	5.758571	0	5.834286	0	0	0	0	5.647143	3.827143	0	5.797143	4.202857143
29.77286	0	0	22.82286	0	22.36429	0	0	0	0	25.63571	18.28	0	22.77429	20.762857143
95.05143	0	0	81.88857	0	67.99571	0	0	0	0	78.40857	45.26571	0	65.06571	48.72428571
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

根据 TextRank 权重表对附件 2 中的所有留言进行分类，取每条留言的最大权重值对应的指标名作为该条留言的标签名，并记分类结果为“TextRank 分类”，最终计算得到 F-Score 的值为 0.8413。

3.3. 分类算法对比

3.3.1. 最近邻算法

K 最近邻（k-Nearest Neighbor，KNN）分类算法，是一个理论上比较成熟的方法，属于机器学习算法中的一种。该方法的思路是：如果一个样本在特征空间中的 k 个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。KNN 算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。因此，在 python 中调用 KNN 算法的时候，通过对 n_neighbors（最邻近的样本个数）的值进行调整，使算法达到最优。

根据 3.2 中构建的权重表以及对应的 F-Score 的值，可以得到基于 TF-IDF 权重表的得分最高,故先基于 TF-IDF 权重表展开分析。按随机种子 0 将附件 2 中的留言分为训练集和测试集，通过对比发现，当 n_neighbors=5，即值为默认值时，算法训练拟合的效果最好，其中训练集的 score 为 0.89，测试集的 score 为 0.84。根据训练集与测试集的得分，可以确定该算法存在过拟合的现象，因此需要通过更换随机种子训练出更适用的算法。当选择随机种子的值为 13 时，训练集的 score 为 0.88，测试集的 score 为 0.86。利用该算法对附件 2 中所有留言展开分类，记分类结果为“最近邻 TF-IDF 分类”，得到总的 F-Score 的值为 0.8729，各个分类的 F-Score 的值如表 6 所示：

表 6 最近邻算法分类得分表

	TP	FP	FN	precision	recall	F1
城乡建设	1692	318	317	0.8418	0.8422	0.8420
党务政务	0	0	0			
国土资源	0	0	0			
环境保护	865	83	73	0.9124	0.9222	0.9173
纪检监察	0	0	0			
交通运输	525	146	88	0.7824	0.8564	0.8178
经济管理	0	0	0			
科技与信息产业	0	0	0			
民政	0	0	0			
农村农业	0	0	0			
商贸旅游	949	163	266	0.8534	0.7811	0.8156
卫生计生	759	78	118	0.9068	0.8655	0.8856
政法	0	0	0			
教育文体	1465	122	124	0.9231	0.9220	0.9225
劳动和社会保障	1825	220	144	0.8924	0.9269	0.9093

基于词频权重表，采用相同参数训练拟合最近邻算法，接着利用该算法对附件 2 中所有留言进行分类，记分类结果为“最近邻词频分类”，得到总的 F-Score 的值为 0.8553。同理，基于 TextRank 权重表拟合模型分类，记分类结果为“最近邻 TextRank 分类”，得到总的 F-Score 的值为 0.8656。

3.3.2. 随机森林算法和梯度提升算法

随机森林（Random Forest）是一种包含多个决策树的分类器。对于每棵树，使用的训练集是从总的训练集中采用有放回的方式采样出来的。而在训练每棵树的结点时，使用的特征是从所有特征中按照一定比例采用无放回的方式随机抽取的。

采取最近邻算法模型训练中相似的方法，使用随机种子将附件 2 中的留言分为训练集和测试集，并在 python 中调用随机森林算法，通过调节决策树的数量，得到当 n_estimators=4 时，训练拟合的效果相对是最优的，其中训练集的 score 为 0.96，测试集的 score 为 0.78。由此可见，利用随机森林算法进行拟合分类存在过拟合的情况。

梯度提升（Gradient Boosting Decision Tree，GBDT）算法的核心就是用损失函数的负梯度在当前模型 F(x)下的取值，来近似拟合残差，即

$$-\left[\frac{\partial L(y^{(i)},F(x^{(i)}))}{\partial F(x^{(i)})}\right]_{F(x^{(i)})=F_{k-1}(x^{(i)})} \tag{4}$$

因此在 python 中调用梯度提升算法，通过调节各参数值，得到训练拟合效果相对最优的状况下，其训练集的 score 为 0.92，测试集的 score 为 0.85。因此，可以发现在利用梯度提升算法进行拟合分类时也存在过拟合的情况。

3.3.3. 支持向量机算法

支持向量机（Support Vector Machine，SVM）将向量映射到一个更高维的空间里，在这个空间里建立有一个最大间隔超平面。此外，在机器学习模型中，需要人工选择的参数称为超参数。如果超参数的选择不恰当，则会导致出现过拟合或者欠拟合的情况。而在选择超参数的时候，有两种方法，其中一种是结合实际进行微调，另一种是将不同大小的参数带入到相应模型中，从而挑选出表现最好的参数。因此使用 Scikit-Learn 中的 GridSearchCV 来完成这项搜索工作，即通过网格搜索选取最优的超参数。

当选择随机种子的值为 13 时，该算法选择的最优参数：“C”参数值为 100、“gamma”参数值为 0.0001，其训练集的 score 为 0.89，测试集的 score 为 0.88。利用该算法对附件 2 中所有留言展开分类，记分类结果为“支持向量机 TF-IDF 分类”，得到总的 F-Score 的值为 0.8825。通过调整随机种子进行对比，得到当选择随机种子的值为 15 时，该算法选择的最优参数：“C”参数值为 100、“gamma”参数值为 0.0001，其训练集的 score 为 0.89，测试集的 score 为 0.89。利用该算法对附件 2 中所有留言展开分类，记分类结果为“支持向量机 TF-IDF

分类”，得到总的 F-Score 的值为 0.8836。

3.4. 留言群划分

3.4.1. 基于最近邻算法

目前大部分电子系统依靠人工根据经验对留言进行处理，存在出错率高、工作量大且效率低的问题，故需对留言分类展开进一步处理。根据前述分析，基于 TF-IDF 权重表，利用最近邻算法进行分类，效果最好，但该分类结果与实际分类存在一定出入，即对于留言仍需人工处理一遍。因此，在实际分类中，为帮助公司更合理的安排人员，减少工作量以及提高工作效率，可以利用程序先展开初步分类筛选，将留言群大致分为两类，一类留言群需要人工进一步分析处理，另一类留言群可使用程序添加标签名。

因为在利用最近邻算法的情况下，基于 TF-IDF 权重表进行分类效果最好，故以该分类结果作为分类参考，即以“最近邻 TF-IDF 分类”指标作为参考指标。首先根据 3.2 中构建的“最近邻 TF-IDF 分类”和“最近邻词频分类”这两个指标，若某条留言所对应的这两个指标值不同，即两种方法的分类结果不同，则使该留言归属于留言群 1，这类留言数量大约占总留言数的 13%。对于该留言群，通过将添加的分类标签名与实际分类标签名进行对比，得到基于“最近邻 TF-IDF 分类”指标的 F-Score 值为 0.5143，基于“最近邻词频分类”指标的 F-Score 值为 0.3885。故对于该类留言群中的留言，需人工再次进行仔细审核分类。

其次对于不属于留言群 1 的留言，查看留言对应的“最近邻 TF-IDF 分类”指标和“最近邻 TextRank 分类”指标，若这两个指标值不同，则使留言归属于留言群 2，这类留言数量大约占总留言数的 4%。对于该留言群，通过将添加的分类标签名与实际分类标签名进行对比，得到基于“最近邻 TF-IDF 分类”指标的 F-Score 值为 0.7135，基于“最近邻 TextRank 分类”指标的 F-Score 值为 0.2290。故对于该类留言群中的留言，可参考“最近邻 TF-IDF 分类”的指标值，但需人工再次进行仔细审核分类。

接着对于不属于留言群 1 和留言群 2 的留言，查看留言对应的“最近邻 TF-IDF 分类”指标和在 3.2 中根据最大值构建的“TF-IDF 分类”指标，若这两个指标值不同，则使留言归属于留言群 3，这类留言数量大约占总留言数的 4%。对于该留言群，通过将添加的分类标签名与实际分类标签名进行对比，得到基于“最近邻 TF-IDF 分类”指标的 F-Score 的值为 0.5324，基于“最近邻 TextRank 分类”指标的 F-Score 的值为 0.2189。故对于该类留言群中的留言，需人工再次进行仔细

审核分类。

除去已划分至留言群的留言，剩余留言“最近邻 TF-IDF 分类”、“最近邻词频分类”、“最近邻 TextRank 分类”和“TF-IDF 分类”这四个指标的值均相同。如果某条留言满足在 TF-IDF 权重表、词频权重表和 TextRank 权重表中的最大权重值均不大于 10，则归属于留言群 4，该类留言群的留言数量大约占总留言数的 1%，得到该留言群分类的 F-Score 的值为 0.8448。这部分留言的留言主题与留言详情的内容相对较简短，存在易识别分类出错的情况，故可参考“最近邻 TF-IDF 分类”的指标值，但需人工再次审核分类。

最后剩下的所有留言归属于留言群 5，该类留言群的留言数量大约占总留言数的 78%。通过将添加的分类标签名与实际分类标签名进行对比，得到 F-Score 的值为 0.9499，其中各个分类的 F-Score 的值如表 7 所示：

表 7 留言群 5 分类得分表

	TP	FP	FN	precision	recall	F1
城乡建设	1295	62	89	0.9543	0.9357	0.9449
党务政务	0	0	0			
国土资源	0	0	0			
环境保护	805	30	14	0.9641	0.9829	0.9734
纪检监察	0	0	0			
交通运输	422	55	17	0.8847	0.9613	0.9214
经济管理	0	0	0			
科技与信息产业	0	0	0			
民政	0	0	0			
农村农业	0	0	0			
商贸旅游	718	35	88	0.9535	0.8908	0.9211
卫生计生	673	29	30	0.9587	0.9573	0.9580
政法	0	0	0			
教育文体	1364	73	31	0.9492	0.9778	0.9633
劳动和社会保障	1557	45	60	0.9719	0.9629	0.9674

进一步分析错分的留言，可以发现留言群 5 中大部分错分的留言内容较模糊，既可归属于原有的一级标签，也可归属于新划分的标签，故对于留言群 5，由模型所添加的标签名可信度高，可直接使用，无需人工再次审核分类。

至此，所有留言都已划分至具体的留言群，划分的留言群信息如表 8 所示：

表 8 基于最近邻算法的留言群

类别（占比）	分类指标	F-Score	分类指标	F-Score	分类方式
留言群 1（13%）	最近邻 TF-IDF 分类	0.5143	最近邻 词频分类	0.3885	人工审核分类
留言群 2（4%）	最近邻 TF-IDF 分类	0.7135	最近邻 TextRank 分类	0.2290	人工参考审核分类
留言群 3（4%）	最近邻 TF-IDF 分类	0.5324	TF-IDF 分类	0.2189	人工审核分类
留言群 4（1%）	最近邻 TF-IDF 分类	0.8448			人工参考审核分类
留言群 5（78%）	最近邻 TF-IDF 分类	0.9499			自动分类（“最近邻 TF-IDF 分类”指标）

3.4.2. 基于支持向量机算法

当选择随机种子为 15 时，利用支持向量机算法进行分类，基于 TF-IDF 权重表的分类效果最好，故以该分类结果作为分类参考，即以“支持向量机 TF-IDF 分类”指标作为参考指标。得到五类留言群的 F-Score 情况如表 9 所示：

表 9 基于支持向量机算法的留言群

类别（占比）	分类指标	F-Score	分类指标	F-Score	分类方式
留言群 1（9.5%）	支持向量机 TF-IDF 分类	0.5718	支持向量机 词频分类	0.3478	人工审核分类
留言群 2（2.5%）	支持向量机 TF-IDF 分类	0.7067	支持向量机 TextRank 分类	0.2464	人工参考审核分类
留言群 3（5.5%）	支持向量机 TF-IDF 分类	0.5442	TF-IDF 分类	0.2031	人工审核分类
留言群 4（1%）	支持向量机 TF-IDF 分类	0.7629			人工参考审核分类
留言群 5（81.5%）	支持向量机 TF-IDF 分类	0.9404			自动分类（“支持向量机 TF-IDF 分类”指标）

通过对比基于最近邻算法的结果，可以发现基于支持向量机算法比基于最近邻算法的留言群 5 的占比高，但基于支持向量机算法比基于最近邻算法的留言群 5 的 F-Score 值低了将近一个百分点。其余留言群的 F-Score 值情况也各有优劣，故接下来仍从两种算法出发，分别构建模型，继而通过对比选取最优模型。

3.5. 灵敏性分析

通过调整在构建权重表时赋予的权重值，调用最近邻算法进行训练拟合产生新模型，并与原最近邻算法的训练集得分和测试集得分进行对比，从而检验权重值的可靠性。

将 a 的值设为 9，b 和 c 的值仍为 5 和 1，据此重新构建基础权重表，并利用 TF-IDF 算法提取的关键词进行匹配，对基础权重表进行填充得到新的 TF-IDF 权重表。接着采用最近邻算法进行训练拟合，得到训练集的 score 仍为 0.88，测试集的 score 仍为 0.86。使用拟合后的算法对所有留言进行分类，得到总的 F-Score 的值为 0.8727，相较原来的 0.8729 有所降低，但基本相同，故权重值的设定可靠度高，可以采用。

对于关键词的权重值设定的可靠性分析，先按照关键词 TF-IDF 值、TextRank 值或词频数由大到小的顺序将关键词的权重值依次设为 1～1/6 的一串等差数值。接着基于基础权重表，重新构建 TF-IDF 权重表、词频权重表和 TextRank 权重表，然后分别使用最近邻算法训练模型，计算不同权重表对应训练集和测试集的 F-Score，可以发现 F-Score 变动不大。以新的 TF-IDF 权重表为例，采用最近邻算法展开训练分类，得到训练集的 score 为 0.89，测试集的 score 仍为 0.86，而总的 F-Score 的值为 0.8736，较原来的 0.8729 有所提高，但基本相同，故权重值的设定可靠度高，可以采用。

3.6. 分类模型构建

3.6.1. 模型建立（基于最近邻算法）

基于上述分析，采用最近邻算法可以使留言的分类效果达到最优。同时，可将多种权重表进行组合，得到五类留言群，并根据每类留言群的特性，给予不同的处理，从而达成减轻工作量、提高效率的目标。

对于新留言，先使用标签名进行匹配，构建基础权重表。再分别使用 3.2 中通过 TF-IDF 算法、词频统计和 TextRank 算法提取的相应关键词，实现权重的二次填充，得到 TF-IDF 权重表、词频权重表和 TextRank 权重表。基于这三张权重表，然后分别采用 3.3.1 中对应权重表的最近邻算法进行分类，记为“最近邻

TF-IDF 分类”、“最近邻词频分类”和“最近邻 TextRank 分类”，并提取 TF-IDF 权重表中每条留言的最大权重值对应的指标名作为该条留言的“TF-IDF 分类”指标值。根据新构建的四个分类指标，以及每条留言在三张权重表中的最大权重值，先将留言划分到对应的留言群中，最后输出留言被划分到的留言群以及为其划分的一级标签。

根据上述流程，建立分类模型，例如对于如表 10 中两个留言：

表 10 留言分类样例

留言编号	留言用户	留言主题	留言时间	留言详情	一级标签
2603	A00099650	请调查西地省建望集团及西地省辉东安建工程有限公司的违法行为	2019/4/20 16:50:49	我叫涂愈，是一名普通的建筑业从业者，在这里求助！也请求相关部门调查西地省建望（集团）及西地省辉东安建工程有限公司涉黑涉恶的违法行为！事情是这样的，2012年我通过了二级建造师执业资格考试，并在2013年取得二级建造师执业资格证书（见附图）。一次在招聘网站上看到西地省建望（集团）有限公司的招聘建造师信息后，投了简历，随后接到了通知，叫我带上身份证、学历证以及二级建造师执业资格证去面试。而后，他们给了一份建造师聘用合同（见附图），让我同意在上面签字，说先签了这个，再等公司通知来上班。后来一直没有接到上班的通知，我才明白，他们要的不是我去他们那里上班，只是需要用我的证书来取得他们公司的什么资质，既然已经签字了，我也就认了…可是，他们也没有按签订好的建造师聘用合同付款，我记得第一年（2013年～2014年）的费用是拖了两个月时间才付，而后第二年（2014年～2015年）就拖了一年多才付的，第三年（2015年～2016年）就至今没有付给我了（而且每次要提供相应金额的发票才能办付款手续）。后来我才查询到这种行为就是建筑行业通常所说的“挂靠”行为，是一种违法的行为，这钱我也就不好去要了，只好忍气吞声，有苦也说不出。去年8月，我从外地回到A市，心里想到我的证也到期了，应该可以拿出来了，顺便再问问他们能不能把（2015年～2016年）的费用给我不，8月24日我去了建望（集团）公司找到了一位姓周的主任，跟她说明了我的情况，她告知我的证已经转到别的公司去了，费用的问题说要耐心等待。我一听到这个消息就懵了，我的证在你们公司怎么会无缘无故在别的公司了，后来在网站上的确查到我的证书注册在一个叫西地省辉东安建工程有限公司名下（见附图）！我就纳闷了，没有我的签字，也没有经过我同意，证书怎么会从建望（集团）公司注册到别的公司去了！我当天就去这家叫西地省辉东安建工程有限公司的办公室，他们办公室人员说不知情，一口否定！我心中的怨气真的不知道怎么形容…这几天，我思前想后，心里越来越不对劲，因为我势单力薄，只是一个外来到A市务工的人员，法律意识浅薄，我只好向领导求助！希望法律是可以保护我们这些弱势群体的，希望通过有关部门的调查可以还我一个公道！跪求！西地省建望（集团）有限公司或存在以下违法行为：1、诱骗他人签订违背本人意愿合同行为；2、“挂证”违法违规行为；3、“偷税漏税”违法行为；4、违规扣押证书行为；5、给其它企业提供注册证书骗取国家机关违规延续注册行为；西地省辉东安建工程有限公司或存在以下违法行为：1、伪造聘用劳动合同违法行为；2、伪造他人签名骗取国家机关违规注册行为；3、冒用身份信息行为；4、“挂证”违法违规行为；	城乡建设
4361	U0003244	长株潭城铁观沙岭到先锋能否增加车次，保证上班族通勤	2018/10/22 17:01:50	我是普通的上班族，住在A2区先锋新宇小区，在A3区银盆岭绿地中央广场上班，原本长株潭城铁开通时，我可以坐城铁上下班，从先锋站到观沙岭站，26分钟到达，方便快捷，但不知从什么时候开始，原本18:26分从观沙岭到先锋站的车次取消了，这让我每天下班只能挤公交、再转地铁，实在是很不方便。据我所知，先锋站到观沙岭的乘客很多，每次都会上很多人。现在晚上6点以后从观沙岭到先锋的车只有20:48分的了，已经太晚了，满足不了通勤的要求。请求铁广公司能否增加6点到7点从观沙岭站到先锋站的城铁，谢谢！	城乡建设

通过调用程序（见附录），得到第一条留言属于留言群 1，故对于该类留言需要人工再次审核分类，实际添加的一级标签为：“劳动和社会保障”，与原本的一级标签“城乡建设”不同，确实需要人工将可能存在的错误分类标签进行修正。第二条留言则属于留言群 5，新添加的一级标签为：“城乡建设”，与原一级标签相同，即分类正确。

根据 3.4 中留言群的构建与分析，对于留言群 1（留言数占总留言数的 13%）和留言群 3（留言数占总留言数的 4%）中的留言，需要人工审核分类；对于留言群 2（留言数占总留言数的 4%）和留言群 4（留言数占总留言数的 1%）中的留言，也需要人工审核分类，但可参考“最近邻 TF-IDF 分类”指标的值，该指标值为基于 TF-IDF 权重表，利用最近邻算法得到的分类结果；对于留言群 5（留言数占总留言数的 78%）中的留言，可直接采用对应的“最近邻 TF-IDF 分类”指标的值，无需人工再次审核分类。最后根据上述分析，将“最近邻 TF-IDF 分类”指标值作为最终分类结果，给所有留言添加上分类标签，总的 F-Score 的值为 0.8729。

3.6.2. 模型优化

综合之前所有的分类方法分别对留言群进行分类，即基于不同的权重表，采

用最近邻算法或支持向量机算法对留言进行分类。然后选取每类留言群中F-score值最高的分类结果进行组合，从而构建最终的分类模型。

首先基于最近邻算法，选择随机种子的值为13，并根据“最近邻TF-IDF分类”、“最近邻词频分类”、“最近邻TextRank分类”和“TF-IDF分类”这四个指标进行留言群的划分。然后以“最近邻TF-IDF分类”指标的值作为初始的一级标签，通过计算对比得到F-Score的值如表11所示：

表11 基于最近邻算法的F-Score表

	留言群1	留言群2	留言群3	留言群4	留言群5	总
分类指标	最近邻TF-IDF分类	最近邻TF-IDF分类	最近邻TF-IDF分类	最近邻TF-IDF分类	最近邻TF-IDF分类	基础模型
F-Score	0.5143	0.7135	0.5324	0.8448	0.9499	0.8729
分类指标	支持向量机TF-IDF分类	最近邻TF-IDF分类	最近邻TF-IDF分类	最近邻TF-IDF分类	最近邻TF-IDF分类	优化模型
F-Score	0.6384	0.7135	0.5324	0.8448	0.9499	0.8904

接着基于支持向量机算法，选择随机种子的值为15，并根据“支持向量机TF-IDF分类”、“支持向量机词频分类”、“支持向量机TextRank分类”和“TF-IDF分类”这四个指标进行留言群的划分。然后以“支持向量机TF-IDF分类”指标的值作为初始的一级标签，通过计算对比得到F-Score的值如表12所示：

表12 基于支持向量机算法的F-Score表

	留言群1	留言群2	留言群3	留言群4	留言群5	总
分类指标	支持向量机TF-IDF分类	支持向量机TF-IDF分类	支持向量机TF-IDF分类	支持向量机TF-IDF分类	支持向量机TF-IDF分类	基础模型
F-Score	0.5718	0.7067	0.5442	0.7629	0.9404	0.8836
分类指标	最近邻TF-IDF分类	支持向量机TF-IDF分类	最近邻词频分类	最近邻TextRank分类	支持向量机TF-IDF分类	优化模型
F-Score	0.6034	0.7067	0.5893	0.8636	0.9404	0.8861

通过比较可以发现，基于最近邻算法的优化模型的总的F-Score的值大于基于支持向量机算法的优化模型的总的F-Score的值。因此，选取随机种子的值为13，并基于3.2中构建的权重表，分别调用最近邻算法和支持向量机算法进行分类。然后根据“最近邻TF-IDF分类”、“最近邻词频分类”、“最近邻TextRank分类”和“TF-IDF分类”这四个指标对留言进行划分。最后留言群1中的留言

以“支持向量机 TF-IDF 分类”的指标值作为一级标签，留言群 2、留言群 3、留言群 4 和留言群 5 中的留言以“最近邻 TF-IDF 分类”的指标值作为一级标签，得到总的 F-Score 的值为 0.8904，其中留言群 5 占比 78%，F-score 的值为 0.9499。

4. 问题二求解

4.1. 问题分析

本题要求先对同类问题进行分类，再根据类群对问题的热度作出判断。因此首先从事发地点入手，先地区，后小区进行特征构造。再利用保留高频词的方法对头部热点问题进行捕捉。而后，以类作为单位，综合考虑回复时间跨度，点赞反对数，以及某个热点问题的样本容量三个维度，利用熵值法+优劣解距离法（TOPSIS）方法进行评价。

4.2. 头部热点问题筛选模型构建

热点问题的描述尽管五花八门，但群众一定会对问题发生的地点有一个详细的刻画，锁定了问题发生的地点，就能大致确定某一类或几类问题，因此利用留言标题中的时间和空间信息对同类热点问题进行聚类。

在实际生活中，热点问题发生的地点一般是确定的，相似问题出现的频率较高且需要一定的解决时间，因此可以假设在一定的时间段内某个地区的热点问题并没有改变，所以，在一个不太长的时间跨度内，完成了对问题地域信息的分类，即完成了热点问题的分类问题。

接着以类为单位，综合考虑回复时间跨度、点赞反对数以及某个热点问题的样本容量这三个维度，利用熵值法+优劣解距离法（TOPSIS）方法进行评价。

4.2.1. 时空划分

使用正则表达式，提取省、市、区、县四类分级指标。同时，按月将数据进行分类，期以观测各地区意见数量的稳定性。

在附件三中的 4326 条观测中，留言的标题与具体内容中均没有出现地点信息的留言数量仅有 73 条。通过观察分析，得到这 73 条数据并没有集中反映某类问题，所以有理由相信去除这 73 条数据不会对定义头部热点问题造成偏差。表 13 是部分市区，在各个月份收到的意见条数，完整的统计表见附件“时间地域分析.xlsx”。

表 13 留言数量表

	a6 区	a3 区	a4 区	a 市	b 市	l 市	西地省
201706	0	0	0	1	0	0	1
201805	0	0	0	1	0	0	0
201810	0	1	0	1	0	0	0
201811	1	0	0	1	0	0	0
201901	14	39	18	240	3	1	68
201902	10	18	19	173	3	1	36
201903	15	46	33	227	2	2	65
201904	30	49	33	235	1	1	72
201905	14	40	30	242	1	1	61
201906	17	46	22	218	1	2	44
201907	20	46	32	272	4	0	59
201908	24	44	39	241	1	0	45
201909	26	59	23	231	2	0	69
201910	12	42	21	174	1	0	43
201911	10	40	20	159	2	0	39
201912	21	46	24	230	0	4	61
202001	3	8	7	51	1	1	7

从时间上看，居民的留言主要集中在 2019 年。2017、2018 年的数据量较少，2020 年 1 月的留言数也不高，这可能是由于只统计了 1 月部分天数的留言信息。从地域上看，省、市这些反映行政区域概念的词与留言的多少并没有明显的相关关系，例如 b 市，l 市的留言数量就明显小于 a 市的某个区或者县，数据中出现了大量包含例如“a 市 a2 区”留言的情况，这里认为 ai 区明显从属于 a 市。

基于上述分析，在空间上，排除月度数据均值小于 10 的县市区，将热点问题的范围缩小到 a 市下的各个辖区之中。同时，也将西地省的组别纳入考虑之中，以此来抓取涉及省级层面的热点问题；在时间上，90%的数据集中在 2019 年。在后续的研究中，可以发现这一年间同地域的热点问题并没有发生改变，因此不用在时间上对数据再进行切分。

4.2.2. 热点问题聚类

前述按照行政区域划分留言的方法初步降低了模型的复杂性。接下来对各组

中的数据按照街道、公路、小区、学校名进一步划分。用正则表达式匹配以‘街道’‘小区’‘社区’‘路’‘公寓’‘学院’等结尾的指示地点的词，保留这些词前面的两个文字或字符，作为位置信息。由于模型目标是捕捉头部的热点问题，因此只保留词频高于千分之三的词，同时为解决正则表达式匹配带来的异常信息问题，需要删除例如‘我走路’‘在 a 市’这样的异常词汇。表 14 是经由正则表达式筛选后的高频“地点”。

表 14 高频地域初步筛选表

地域	频数
东六路	20
丽发新城	53
人行道	18
国际城	24
安置小区	24
暮云街道	14
有限公司	75
楚龙街道	19
泉塘街道	19
消防通道	23
滨河苑	41
物业公司	15
经开区	22
经济学院	14
魅力之城	29
麻将馆	38

进一步分析，发现表中除了有表示地域的词组外，还混入了一些表示问题的对象的词组，例如“麻将馆”、“人行道”、“消防通道”。以麻将馆为例，尽管有许多关于“麻将馆”的留言，但那些麻将馆分布在 a 市的各个地区，政府部门无法通过直接处理某个麻将馆，解决关于麻将馆的所有留言反馈问题。因此，并不认为像“麻将馆”这样表示对象的词组，反映了某一热点问题。

因此，如表 15 所示，我们保留如下 9 类，完成对头部热点问题的聚类。

表 15 高频地域最终筛选表

地域	频数
东六路	20
丽发新城	53
国际城	24
暮云街道	14
楚龙街道	19
泉塘街道	19
滨河苑	41
经济学院	14
魅力之城	29

在上述地名类群中，尽管也存在着其他不同方面的问题，但绝大多数的问题反映了同一热点。我们认为反映其他问题的噪声在各个类中均有存在，通过建立有效的评价指标，这类噪声并不会给热点问题的分类造成特别重大的影响。因此，在分类中，对这种类间的噪声不进行处理。

4.3. 热点问题评价

对于上述初步选出的九大问题地区，综合考虑回复时间跨度，点赞反对数以及某个热点问题的样本容量这三个维度，利用熵值法+优劣解距离法（TOPSIS）方法进行评价，选取热度最高的 5 个热点问题。

首先建立三大指标：

留言时间跨度，该指标反映了该问题的存在时间，该指标越大，则说明当前问题的热度越高，越应当早日解决。对于某类问题的留言时间跨度指标，

留言时间跨度=最新留言时间-首条留言时间

点赞反对数，该指标反映了其它群众作为一个旁观者，对这条留言的认可度，一定程度上描述了该问题在民间的受关注程度。对于某类问题，

点赞反对数=点赞数-反对数的类间平均值

样本容量，该指标统计了反映同一类问题的居民人数，最直接地反映了某一问题的热度，某类问题的样本容量指标等于该类问题的观测数目。

由于上述数据考察的维度不同，量纲不同，使用熵值法，根据方差中的信息，对指标进行赋权。各类问题的原始指标数据如表 16 所示：

表 16 各类问题原始指标数据表

地域	delta_time	vote_num	count
东六路	330	7.55	20
丽发新城	291	5.09434	53
国际城	331	0.291667	24
暮云街道	364	0.071429	14
楚龙街道	338	1.315789	19
泉塘街道	308	3.157895	19
滨河苑	45	0.536585	41
经济学院	902	-0.85714	14
魅力之城	330	0.068966	29

各指标具体权重如表 17 所示：

表 17 指标权重表

指标	权重
delta_time	0.302825
vote_num	0.351817
count	0.345357

接下来使用 TOPSIS 方法对各类问题进行打分，充分利用原始数据的信息，精确地反映各类问题之间的热度差异常。

最终得到如表 18 所示结果：

表 18 各类问题得分表

地域	得分	地域	得分
东六路	0.1909	泉塘街道	0.1197
丽发新城	0.1881	滨河苑	0.0818
国际城	0.0709	经济学院	0.124
暮云街道	0.0651	魅力之城	0.075
楚龙街道	0.0844		

如表 15 所示，排名前五的热点问题分布在东六路、丽发新城、经济学院、泉塘街道、楚龙街道。具体热点问题描述如表 19 所示：

表 19 热点问题描述表

排名	问题 ID	热度指数	时间范围	地点/人群	问题描述
1	1	0.1909	2019-01-07 至 2019-12-03	A 市 A7 县东六路	东六路规划不明确，存在效率与安全的隐患
2	2	0.1881	2019-04-10 至 2020-01-26	A 市 A2 区丽发新城	小区周围的搅拌站，污染环境，影响生活
3	3	0.124	2017-06-08 至 2019-11-27	A 市经济学院	组织学生强制实习
4	4	0.1197	2019-03-05 至 2020-01-07	A 市万科魅力之城小区	小区临街餐饮店油烟噪音扰民
5	5	0.0844	2019-01-07 至 2019-12-11	A 市伊景园滨河苑	小区捆绑销售车位给业主

5. 问题三求解

5.1. 问题分析

本问要求从答复的相关性、完整性、可解释性等角度出发，构建答复评价模型，从而对相关部门答复意见的质量进行评价。因此，需要明确刻画相应指标的内涵，通过计算得到相应的指标值，然后将所有指标值进行标准化处理，对应相加得到各条答复意见的权重值。最后，根据权重值大小将答复意见划分为四类，即“完美”、“较好”、“达标”、“待改善”。

5.2. 指标构建

5.2.1. “相关性”指标构建

“相关性”指标刻画的是答复意见与整体留言的相关程度，因此考虑从留言主题和留言详情入手，通过 TF-IDF 模型从相关词的词频权重的角度分别考察两者与答复意见间的相关度。对于一条留言数据，首先通过定义正则表达式，去除留言主题与留言详情中相应的正则表达式。接着利用 python 中的 jieba 分词，对留言主题与留言详情分别进行分词处理，并定义去除词库，将不在去除词库中的词存到留言分词列表中。同时将答复意见也进行分词处理，然后基于留言分词列表构建词典，即将留言分词进行编号处理，并利用词典构建语料库。利用构建的

词典把答复意见的分词也转换为二元组的向量，即将答复意见的分词作如下处理：如果答复意见中的某分词在词典中出现，则将该分词换成对应的词典中的编号，并统计该分词在答复意见中的频数，从而构成二元组向量，且记该分词为关键词；如果分词在分词列表中不存在，则忽略。接着根据原始留言分词语料库和答复意见关键词二元组向量，采用 TF-IDF 模型进行建模，计算得到基于 TF-IDF 模型的稀疏向量集与基于 TF-IDF 模型的关键词稀疏向量，而后，将稀疏向量集与关键词稀疏向量中相同编号的值相乘求和得到留言主题与答复意见的相关度，以及留言详情与答复意见的相关度。假设留言主题的重要性为留言详情的一倍，得到“相关性”指标的计算公式如下：

$$\text{“相关性”指标} = 2\times \text{相关度}_{(\text{留言主题与答复意见})} + \text{相关度}_{(\text{留言详情与答复意见})} \quad (5)$$

根据上述方法，给附件 4 中的所有答复构建“相关性”指标。通过分析指标值，得到该指标的最小值为 0，上四分位数为 0.3151，中位数为 0.5512，下四分位数为 0.9487，最大值为 2.0694。为保证指标的规范性，因此，将大于 1 的指标值重新赋值为 1，即该指标的满分为 1，故其指标值越大相关性越高。以表 20 中的留言为例：

表 20 留言-答复表

留言编号	留言用户	留言主题	留言时间	留言详情	答复意见	答复时间
2549	A00045581	A2区景蓉华苑物业管理有问题	2019/4/25 9:32:09	2019年4月以来，位于A市A2区桂花坪街道的A2区公安分局宿舍区（景蓉华苑）出现了一番乱象，该小区的物业公司美顺物业扬言要退出小区，因为小区水电改造造成物业公司的高昂水电费收取不了(原水电在小区买，水4.23一吨，电0.64一度）所以要通过征收小区停车费增加收入，小区业委会不知处于何种理由对该物业公司一再挽留，而对业主提出的新应聘的物业公司却以交20万保证金，不能提高收费的苛刻条件拒之门外，业委会在未召开全体业主大会的情况下，制定了一高昂收费方案要各业主投票，而投票不采用投票箱只制定表格要物业公司人员这一利害关系机构负责组织，对投票业主隐私权没有任何保护，还对投反对票的业主以领导做工作等方式要求改变为同意票，这种投票何来公平公正公开，面对公安干警采用这种方式投票合法性在哪？	现将网友在平台《问政西地省》栏目向胡华衡书记留言反映“A2区景蓉花苑物业管理有问题”的调查核实情况向该网友答复如下：您好，首先感谢您对我们工作的信任和支持，关于您在平台栏目给胡华衡书记留言，反映“A2区景蓉花苑物业管理有问题”的情况已收悉。现将我们调查处理情况答复如下：经调查了解，针对来信所反映的“小区停车收费问题”，景蓉华苑业委会于2019年4月10日至4月27日以“意见收集方式”召开了业主大会，经业委会统计，超过三分之二的业主同意收取停车管理费，在业主大会结束后业委会也对业主提出的意见和建议进行了认真疏理归纳并进行了反馈，业委会制定的停车收费标准不高于周边小区价格。针对来信所反映的“物业公司去留问题”，5月5日下午,辖区桂花坪街道牵头组织社区、物业公司、业主委员会、业主代表的会议，区住房和城乡建设局也参加了会议。在综合各方面的意见后，辖区桂花坪街道、区住房和城乡建设局已要求业委会依法依规召开业主大会，根据业主大会的表决结果再执行相应的程序。再次感谢您对我区工作的理解和关心。2019年5月9日	2019/5/10 14:56:53

首先根据定义的正则表达式，去除留言主题与留言详情中相应的正则表达式，然后对留言主题和留言详情分别进行分词处理，得到分词列表如图 1 所示：

```
[['A2', '区景蓉华苑', '物业管理', '问题'], ['2019', '以来', '位于', '市', 'A2', '区', '桂花',
'坪', '街道', 'A2', '区', '公安分局', '宿舍区', '景蓉华苑', '出现', '一番', '乱象', '小区', '物业公
司', '美顺', '物业', '扬言', '退出', '小区', '因为', '小区', '水电', '改造', '造成', '物业公司',
'高昂', '水电费', '收取', '不了', '水电', '小区', '买水', '4.23', '一吨', '电', '0.64', '一度',
'所以', '通过', '征收', '小区', '停车费', '增加收入', '小区', '业委会', '不知', '处于', '何种', '理
由', '物业公司', '一再', '挽留', '业主', '提出', '应聘', '物业公司', '以交', '20', '保证金', '不
能', '提高', '收费', '苛刻', '条件', '拒之门外', '业委会', '召开', '全体', '业主大会', '情况', '制
定', '高昂', '收费', '方案', '业主', '投票', '投票', '采用', '投票箱', '制定', '表格', '物业公司',
'人员', '这一', '利害关系', '机构', '负责', '组织', '投票', '业主', '隐私权', '没有', '任何', '保
护', '投', '反对票', '业主', '领导', '工作', '方式', '要求', '改变', '同意', '票', '这种', '投票',
'何来', '公平', '公正', '公开', '面对', '公安干警', '采用', '这种', '方式', '投票', '合法性']]
```

图 1 留言分词

根据上图，可以看到前一个列表为留言主题的分词，后一个列表为留言详情的分词。此外，答复意见的分词如图 2 所示：

```
['现将', '网友', '在', '平台', '《', '问政', '西地省', '》', '栏目', '向', '胡华衡', '书记', '留
言', '反映', '“', 'A2', '区景蓉', '花苑', '物业管理', '有', '问题', '”', '的', '调查核实', '情况',
'向', '该', '网友', '答复', '如下', '：', '您好', '，', '首先', '感谢您', '对', '我们', '工作',
'的', '信任', '和', '支持', '，', '关于', '您', '在', '平台', '栏目', '给', '胡华衡', '书记', '留
言', '，', '反映', '“', 'A2', '区景蓉', '花苑', '物业管理', '有', '问题', '”', '的', '情况', '已',
'收悉', '。', '现将', '我们', '调查', '处理', '情况', '答复', '如下', '：', '经', '调查', '了解',
'，', '针对', '来信', '所', '反映', '的', '“', '小区', '停车', '收费', '问题', '”', '，', '景蓉华
苑', '业委会', '于', '2019', '年', '4', '月', '10', '日至', '4', '月', '27', '日以', '“', '意见',
'收集', '方式', '”', '召开', '了', '业主大会', '，', '经', '业委会', '统计', '，', '超过', '三分之
二', '的', '业主', '同意', '收取', '停车', '管理费', '，', '在', '业主大会', '结束', '后', '业委
会', '也', '对', '业主', '提出', '的', '意见', '和', '建议', '进行', '了', '认真', '疏理', '归纳',
'并', '进行', '了', '反馈', '，', '业委会', '制定', '的', '停车', '收费', '标准', '不', '高于',
'周边', '小区', '价格', '。', '针对', '来信', '所', '反映', '的', '“', '物业公司', '去留', '问题',
'”', '，', '5', '月', '5', '日', '下午', '，', '辖区', '桂花', '坪', '街道', '牵头', '组织', '社
区', '、', '物业公司', '、', '业主', '委员会', '、', '业主', '代表', '的', '会议', '，', '区', '住
房', '和', '城乡', '建设局', '也', '参加', '了', '会议', '。', '在', '综合', '各', '方面', '的',
'意见', '后', '，', '辖区', '桂花', '坪', '街道', '、', '区', '住房', '和', '城乡', '建设局',
'已', '要求', '业委会', '依法', '依规', '召开', '业主大会', '，', '根据', '业主大会', '的', '表决',
'结果', '再', '执行', '相应', '的', '程序', '。', '再次', '感谢您', '对', '我区', '工作', '的',
'理解', '和', '关心', '。', '2019', '年', '5', '月', '9', '日']
```

图 2 答复分词

根据留言的分词列表构建词典，得到各分词与编号之间的对应关系如图 3 所示：

```
{'A2': 0, '区景蓉华苑': 1, '物业管理': 2, '问题': 3, '0.64': 4, '20': 5, '2019': 6, '4.23': 7,
'一再': 8, '一吨': 9, '一度': 10, '一番': 11, '不了': 12, '不知': 13, '不能': 14, '业主': 15, '业
主大会': 16, '业委会': 17, '买水': 18, '乱象': 19, '人员': 20, '以交': 21, '以来': 22, '任何': 23,
'位于': 24, '何来': 25, '何种': 26, '保护': 27, '保证金': 28, '停车费': 29, '全体': 30, '公安分
局': 31, '公安干警': 32, '公平': 33, '公开': 34, '公正': 35, '出现': 36, '利害关系': 37, '制定':
38, '区': 39, '反对票': 40, '召开': 41, '合法性': 42, '同意': 43, '因为': 44, '坪': 45, '增加收
入': 46, '处于': 47, '宿舍区': 48, '小区': 49, '工作': 50, '市': 51, '应聘': 52, '征收': 53, '情
况': 54, '所以': 55, '扬言': 56, '投': 57, '投票': 58, '投票箱': 59, '拒之门外': 60, '挽留': 61,
'提出': 62, '提高': 63, '收取': 64, '收费': 65, '改变': 66, '改造': 67, '方式': 68, '方案': 69,
'景蓉华苑': 70, '机构': 71, '条件': 72, '桂花': 73, '水电': 74, '水电费': 75, '没有': 76, '物业':
77, '物业公司': 78, '理由': 79, '电': 80, '票': 81, '组织': 82, '美顺': 83, '苛刻': 84, '街道':
85, '表格': 86, '要求': 87, '负责': 88, '这一': 89, '这种': 90, '退出': 91, '通过': 92, '造成':
93, '采用': 94, '隐私权': 95, '面对': 96, '领导': 97, '高昂': 98}
```

图 3 分词与编号

基于留言分词列表，利用得到的词典建立语料库，得到语料库如图 4 所示：

```
[[(0, 1), (1, 1), (2, 1), (3, 1)], [(0, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10,
1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 4), (16, 1), (17, 2), (18, 1), (19, 1), (20, 1),
(21, 1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1), (29, 1), (30, 1), (31,
1), (32, 1), (33, 1), (34, 1), (35, 1), (36, 1), (37, 1), (38, 2), (39, 2), (40, 1), (41, 1),
(42, 1), (43, 1), (44, 1), (45, 1), (46, 1), (47, 1), (48, 1), (49, 6), (50, 1), (51, 1), (52,
1), (53, 1), (54, 1), (55, 1), (56, 1), (57, 1), (58, 5), (59, 1), (60, 1), (61, 1), (62, 1),
(63, 1), (64, 1), (65, 2), (66, 1), (67, 1), (68, 2), (69, 1), (70, 1), (71, 1), (72, 1), (73,
1), (74, 2), (75, 1), (76, 1), (77, 1), (78, 5), (79, 1), (80, 1), (81, 1), (82, 1), (83, 1),
(84, 1), (85, 1), (86, 1), (87, 1), (88, 1), (89, 1), (90, 2), (91, 1), (92, 1), (93, 1), (94,
2), (95, 1), (96, 1), (97, 1), (98, 2)]]
```

图 4 语料库

利用词典，将答复意见的分词也转换为二元组的向量，结果如图 5 所示：

```
[(0, 2), (2, 2), (3, 4), (6, 2), (15, 4), (16, 4), (17, 5), (38, 1), (39, 2), (41, 2), (43, 1),
(45, 2), (49, 2), (50, 2), (54, 3), (62, 1), (64, 1), (65, 2), (68, 1), (70, 1), (73, 2), (78,
2), (82, 1), (85, 2), (87, 1)]
```

图 5 答复意见分词的二元组向量

根据语料库，采用 TF-IDF 模型进行建模，并获取答复意见中，每个关键词的 TF-IDF 值，结果如图 6 所示：

```
[(2, 0.17277368511627203), (3, 0.34554737023254406), (6, 0.17277368511627203),
(15, 0.34554737023254406), (16, 0.34554737023254406), (17, 0.43193421279068006),
(38, 0.08638684255813601), (39, 0.17277368511627203), (41, 0.17277368511627203),
(43, 0.08638684255813601), (45, 0.17277368511627203), (49, 0.17277368511627203),
(50, 0.17277368511627203), (54, 0.25916052767440806), (62, 0.08638684255813601),
(64, 0.08638684255813601), (65, 0.17277368511627203), (68, 0.08638684255813601),
(70, 0.08638684255813601), (73, 0.17277368511627203), (78, 0.17277368511627203),
(82, 0.08638684255813601), (85, 0.17277368511627203), (87, 0.08638684255813601)]
```

图 6 关键词 TF-IDF 值

进一步计算得到原始留言语料库基于 TF-IDF 模型的稀疏向量集和答复意见关键词基于 TF-IDF 模型的关键词稀疏向量，然后将稀疏向量集与关键词稀疏向量中相同编号的值相乘求和得到留言主题与答复意见的相关度为 0.2993，以及留言详情与答复意见的相关度为 0.4951。因此，“相关性”指标的值为 1.0937，由于该值大于 1 需重设为 1，即该条留言的答复在“相关性”方面的评分为满分。

5.2.2. “完整性”指标构建

对完整性的评价主要考察答复意见的结构完整性，故从尊称、留言主题、回复内容的开头、感谢语和时间这五个角度出发建立“完整性”指标。当在答复意见中出现了对应的结构部分，“完整性”指标的值加 0.2 分，“完整性”指标的满分为 1 分，故该指标值越高，则答复意见的结构越完整，即答复意见的质量越高。

在评价尊称、留言主题、回复内容的开头和感谢语这四个角度时，直接在答复意见中匹配对应的不同内容，对于时间角度，则应先将答复意见进行正则匹配，删除空格，再读取文本，判断最后一个字是否为“日”，以及倒数的第三个或第四个或第五个位置是否为“月”，以此来检验答复意见的最后是否含有日期。各

角度的结构匹配内容具体见表 21。

表 21 结构内容详情

角度	结构内容
尊称	“您好”
留言主题	留言主题或“留言已收悉”
回复内容的开头	“答复如下”或“回复如下”
感谢语	“感谢”或者“谢谢”
时间	检验答复意见的最后是否含有日期

5.2.3. “可解释性”指标构建

相关部门对留言的答复意见，即是对留言内容的相应解释，故答复意见主要是由较多的短句连接形成的具有逻辑性的长叙述。因此短句越多，则解释的越周到。此外，答复意见是针对相关的留言内容而提出的，故答复意见的文本字数与留言详情的文本字数的比值越大，则说明解释的越详细。

由于符号能反应文本的构成，故从符号的角度入手，设逗号和分号的权重为 0.05，句号与感叹号的权重为 0.1，统计答复意见中相应符号的数量并计算对应的累计权重。另外为衡量答复意见和留言详情部分有效文本长度的比值，去除答复意见部分的结构内容，即去除各条答复意见“完整性”指标对应的结构性内容，而留言详情则选择“相关性”指标构建过程中去除正则表达式的留言详情文本。最终构造“可解释性”指标为：

$$\text{“可解释性”指标}=\text{符号累计权重}+\frac{\text{答复意见文本字数}}{\text{留言详情文本字数}} \qquad (6)$$

根据上述方法，给附件 4 中的所有答复构建“可解释性”指标。通过分析指标值，得到该指标的最小值为 0.0050，上四分位数为 1.5069，中位数为 2.7848，下四分位数为 4.5764，最大值为 247.9889。为保证指标的规范性，因此，先将大于 5 的指标值重新赋值为 5，再将所有指标值都除以 5，规范至 0-1 间，即使该指标的满分为 1，故该指标值越大越好。

5.2.4. “时效性”指标构建

答复的及时程度，会影响答复意见的潜在价值，即越及时的答复，能给留言的人们带去更大的帮助，也更能体现出留言的价值与质量。因此，通过计算答复与留言的时间差来衡量对留言的答复是否及时。由于各条答复时间差大小不等，因此需要为“时效性”指标设置初始值为 50，最终“时效性”指标为：

$$\text{“时效性”指标}=\frac{\text{初始值（设为 50）}-\text{时间差}}{50} \quad (7)$$

该指标值的满分为 1 分，其值越大则说明答复的越及时。

5.3. 答复评价模型构建

上述四个指标从不同的角度出发，对答复意见展开了详细的分析处理，在经过处理后，各个指标的值都位于 0-1 区间内。因此，将四个指标值相加最终得到答复意见的评价权重。

基于评价权重，构建答复评价模型，即根据评价权重对答复意见展开如下分类：如果评价权重大于 3，则相应答复意见的质量为“完美”；如果评价权重小于等于 3，但大于 2，则相应答复意见的质量为“较好”；如果评价权重小于等于 2，但大于 1，则相应答复意见的质量为“达标”；如果评价权重小于等于 1，则相应答复意见的质量为“待改善”。对于“相关性”指标构建中作为范例的编号 2549 留言，其各指标数值如表 22 所示：

表 22 答复评价表

相关性	完整性	可解释性	时效性	答复评价
1.0	0.8	0.5120	0.6800	较好

通过调用模型，得到 2816 条答复意见中，得到“完美”评价的有 698 条；得到“较好”评价的有 1402 条；得到“达标”评价的有 588 条；得到“待改善”评价的有 128 条。通过对比答复意见与答复评价，得到该评价结果基本能反应答复意见的质量情况，另外该评价结果也与实际现象较符合。因此，本问中相应指标与模型的构建是可行的，该评价模型可区分出哪些答复需要进行改善，从而给人们提供更好的服务。

6. 模型评价

优点：对于问题一，本文首先根据已给出的三级标签体系作为基础分类关键词，在留言中对此类关键词进行匹配，从而构建基础权重表。由于同类留言信息必然存在一定的相关性，从关键词入手，改进基础权重表，在词频角度构建 TF-IDF 权重表、词频权重表，在词关联度角度构建 TextRank 权重表。通过上述步骤将文本转换为向量，并基于此采用不同的算法展开预测，从而选取最优算法，构建最优的分类模型。接着将留言进行划分，并根据每类留言群的特性，给予不同的处理，从而达成减轻工作量、提高效率的目标。同时，根据划分的留言群实现对

模型的优化，从而提高 F-Score 的值，其中留言群 5 占比 78%，F-score 为 0.9499。对于问题二，从低点着手可以大大简化模型的复杂程度，降低模型的计算量。同时对模型的精度也有较好的把握。对于问题三，从不同角度出发，从而实现对答复意见的全面分析。同时构建合理的指标，以此对答复意见的质量展开更直观的评价。

缺点：对于问题一，权重值的选择不完全客观，因此权重值的选择对分类的结果存在一定的影响。同时，在预测方面，没有进一步深入分析，挖掘如何在避免过拟合的情况下，实现多模型组合预测，从而提高预测的得分。对于问题二，目前省市区多用代号表示，但在实际应用中，可以获得省内各个位置的地名，构造更为合理的位置字典。对于问题三，指标的构建存在一定的主观性，因此对答复意见的评价存在一些瑕疵。

拓展：对于问题一，可以通过更多的对比，从而选择更适合的权重值构建权重表，即寻找会获得更高分类得分的权重值。还可以选择更合适的随机种子，从而训练拟合出更适合的算法，因为不同的随机种子意味着不同的训练集数据，而训练集数据对算法的拟合存在一定的影响。此外，还可以增加提取的关键词数量，因为基于大样本的情况下，关键词数量越多，则可更细致的对留言实现划分，从而提高分类的得分。对于问题二，在实际应用中，省市区名不再是代号，可以从地图中获得详细的地名列表。那么除了头部热点问题以外，对其他问题的热度也可以有一个好的把控。

7. 参考文献

[1] 高茂庭.文本聚类分析若干问题研究.天津大学博士学位论文.2006:13-15.

[2] 谷波，张永奎.文本聚类算法的分析和比较[J].电脑开发与应用，2003,16(11) :4-6.

[3] 李荣路.文本分类及其相关技术研究 [J] .复旦大学.2005.

[4] 吴巧玲.中文分词算法在自然语言处理技术中的研究及应用[J].信息与电脑:理论版，2011(12):39-40.

[5] 邬启为.基于向量空间的文本聚类方法与实现[D].北京交通大学，2014.

[6] 许阳，刘功申，孟魁.基于句中词语间关系的文本向量化算法 [J] .信息安全与通信保密.2014.

[7] 杨丽华，戴齐，郭艳军.KNN 文本分类算法研究[J].微计算机信息

2006.22(21):269-270.

[8] 张培颖.运用有向图进行中文分词研究.[M]计算机工程与应用.2009,45(22).

8. 附录

1.分类模型

```
import numpy as np
import pandas as pd
#需要以下文件在同一个文件夹中：附件1，附件2，以及之前构建的TF-IDF关键词、词频关键词、TextRank关键词、TF-IDF权重表、词频权重表、TextRank权重表
#读取新的留言数据
data = pd.read_excel('分类数据与结果/test.xlsx')
#data = pd.DataFrame([['留言主题','留言详情']],columns=['留言主题','留言详情'])
data_result = pd.read_excel('分类数据与结果/test.xlsx')
#data_result = pd.DataFrame([['留言主题','留言详情']],columns=['留言主题','留言详情'])

data_type = pd.read_excel('分类数据与结果/附件1.xlsx')
#创建权重表
category = data_type.一级分类.unique()
weight = pd.DataFrame(0,index = np.arange(len(data)), columns = category)

#权重填充步骤1：按标签名进行填充
#按一级分类标签对附件1的分类指标进行归类
def datatuple(data):
    data_category = ()
    for temp_category in category:
        #index重置
        temp_data = data_type[data_type.一级分类.isin([temp_category])].reset_index(drop=True)
        data_category = data_category + (temp_data,)
    return data_category

#文本内容选取
for q in range(len(data)):
    text = data.留言主题[q] + data.留言详情[q]

    #分类名称在文本中出现的次数初始化
    data_type['time'] = 0
    #统计留言主题和留言详情对应的标签名的权重
    for i in range(len(data_type)):
        for j in range(3):
            temp = data_type.iloc[i,j]
            if i > 0:
                if temp == data_type.iloc[i-1,j] or temp == '其他':
                    continue
            n = len(temp)
            #标签名匹配
            for k in range(len(text) - n):
                if text[k : k + n] == temp:
                    if j == 0:
                        data_type.loc[i,'time'] = data_type.loc[i,'time'] + 10
                    elif j == 1:
```

```
                    data_type.loc[i,'time'] = data_type.loc[i,'time'] + 5
                else:
                    data_type.loc[i,'time'] = data_type.loc[i,'time'] + 1
    #采用前面构建的分类函数
    data_category = datatuple(data_type)
    #进行权重汇总填充
    for p in range(len(data_category)):
        weight.iloc[q,p] = sum(data_category[p].time)
    #指标处理：若文本中不含有经济则进行权重归0处理
    t = 0
    for k in range(len(text) - 2):
        if text[k : k + 2] == '经济':
            t = 1
            break
    for k in range(len(text) - 1):
        if text[k : k + 1] == '元' or t == 1:
            t = 1
            break
    if t == 0:
        weight.loc[q,'经济管理'] = 0

#权重填充步骤2：按关键词进行填充
def Value(data,keyword):
    new_weight = pd.DataFrame(0,index = np.arange(len(data)), columns = category)
    word = keyword.T.reset_index(drop=True)
    #文本内容选取
    for q in range(len(data)):
        text = data.留言主题[q] + data.留言详情[q]
        #统计留言主题和留言详情对应的关键词的权重
        for i in range(len(category)):
            for j in range(500):
                temp = word.iloc[2*i,j]
                if type(temp) != str:
                    continue
                n = len(temp)
                #关键词匹配
                for k in range(len(text) - n):
                    if text[k : k + n] == temp:
                        new_weight.iloc[q,i] = new_weight.iloc[q,i] + 1 - j/700
    return new_weight

keyword = pd.read_excel('关键词/TF-IDF关键词.xlsx')
weight1 = Value(data,keyword) + weight
array = np.array(weight1)
data_result['TF-IDF分类'] = category[np.argmax(array,axis=1)]
data_result['TF-IDF分类权重'] = array.max(axis=1)

keyword = pd.read_excel('关键词/词频关键词.xlsx')
weight2 = Value(data,keyword) + weight
array = np.array(weight2)
data_result['词频分类'] = category[np.argmax(array,axis=1)]
data_result['词频分类权重'] = array.max(axis=1)

keyword = pd.read_excel('关键词/TextRank关键词.xlsx')
weight3 = Value(data,keyword) + weight
array = np.array(weight3)
data_result['TextRank分类'] = category[np.argmax(array,axis=1)]
data_result['TextRank分类权重'] = array.max(axis=1)
```

```
#读取附件 2 留言信息与 TF-IDF 权重表，训练拟合最近邻算法
data_old = pd.read_excel('分类数据与结果/附件 2.xlsx')
weight = pd.read_excel('权重表/TF-IDF 权重表.xlsx')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(weight, data_old['一级标签'], ran
dom_state = 13)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
#对新留言进行分类
data_result['最近邻 TF-IDF 分类'] = knn.predict(weight1)

from sklearn.svm import SVC
svc = SVC(kernel='rbf', C=100, gamma=0.0001, probability=True)
svc.fit(X_train, y_train)
data_result['支持向量机 TF-IDF 分类'] = svc.predict(weight1)

#读取词频权重表，训练拟合最近邻算法
weight = pd.read_excel('权重表/词频权重表.xlsx')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(weight, data_old['一级标签'], ran
dom_state = 13)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
#对新留言进行分类
data_result['最近邻词频分类'] = knn.predict(weight2)

#读取 TextRank 权重表，训练拟合最近邻算法
weight = pd.read_excel('权重表/TextRank 权重表.xlsx')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(weight, data_old['一级标签'], ran
dom_state = 13)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
#对新留言进行分类
data_result['最近邻 TextRank 分类'] = knn.predict(weight3)

#留言群的划分
for i in range(len(data_result)):
    if data_result.loc[i,'最近邻 TF-IDF 分类'] != data_result.loc[i,'最近邻词频分类']:
        data.loc[i,'留言群'] = '留言群 1'
    elif data_result.loc[i,'最近邻 TF-IDF 分类'] != data_result.loc[i,'最近邻 TextRank 分
类']:
        data.loc[i,'留言群'] = '留言群 2'
    elif data_result.loc[i,'最近邻 TF-IDF 分类'] != data_result.loc[i,'TF-IDF 分类']:
        data.loc[i,'留言群'] = '留言群 3'
    elif data_result.loc[i,'TF-IDF 分类权重'] < 10 and data_result.loc[i,'词频分类权重']
 < 10 and data_result.loc[i,'TextRank 分类权重'] < 10:
        data.loc[i,'留言群'] = '留言群 4'
    else:
        data.loc[i,'留言群'] = '留言群 5'
    data.loc[i,'一级分类标签'] = data_result.loc[i,'最近邻 TF-IDF 分类']
```

```
Index = data[data['留言群']=='留言群 1'].index
data.loc[Index,'一级分类标签'] = data_result.loc[Index,'支持向量机 TF-IDF 分类']

#print(data)
data.to_excel('分类数据与结果/test 分类.xlsx',index=False)
```

3. 答复评价模型

```
import math
import numpy as np
import pandas as pd
import re # 正则表达式库
import jieba  #引入分词 API 库 jieba
from gensim import corpora,models,similarities   #文本相似度库 gensim

#读取数据
data = pd.read_excel('附件 4.xlsx')
data['留言时间'] = pd.to_datetime(data['留言时间'])
data['答复时间'] = pd.to_datetime(data['答复时间'])
#构建答复指标汇总表
reply = pd.DataFrame(0,index = np.arange(len(data)), columns = ['相关性','完整性','可
解释性','时效性'])

for i in range(len(data)):
    suggestion = data.答复意见[i]
    #相关性指标构建,选择留言主题与留言详情
    text1 = data.留言主题[i]
    text2 = data.留言详情[i]

    all_text = []
    all_text.append(text1)
    all_text.append(text2)
    #对留言主题与留言详情进行分词处理，并且保存在列表 all_text_list 中
    all_text_list = []
    for text in all_text:
        pattern = re.compile(u'\t|\n|，|。|-|：|；|\(|\)|）|（|？|！|“|\xa0|\r| |  ')
        text = re.sub(pattern, '', text)  #将符合模式的字符去除
        text_list = [word for word in jieba.cut(text)]
        object_list = []
        remove_words = list(open('stopwords.txt','r',encoding='gbk').read())
        for word in text_list:
            if word not in remove_words:
                object_list.append(word)
        all_text_list.append(object_list)
    #对答复意见进行分词
    suggestion_list = [word for word in jieba.cut(suggestion)]
    #用 dictionary 方法获取词袋（bag-of-words）
    dictionary = corpora.Dictionary(all_text_list)
    #使用 doc2bow 制作语料库
    corpus = [dictionary.doc2bow(doc) for doc in all_text_list]
    #把答复意见的分词也转换为二元组的向量
    suggestion_vec = dictionary.doc2bow(suggestion_list)
    #使用 TF-IDF 模型对语料库建模
    tfidf = models.TfidfModel(corpus)
    #留言与答复意见的相关性值=TF-IDF 模型的稀疏向量集*TF-IDF 模型的关键词稀疏向量
```

```
    index = similarities.SparseMatrixSimilarity(tfidf[corpus], num_features=len(dict
ionary.keys()))
    sim = index[tfidf[suggestion_vec]]
    #计算留言与答复的相关性
    relativity = 2 * sim[0] + sim[1]
    if relativity > 1:
        relativity = 1
    reply.loc[i,'相关性'] = reply.loc[i,'相关性'] + relativity

    #完整性指标构建，去除答复意见中的空格
    pattern = re.compile(u'\t| |  ')
    suggestion = re.sub(pattern, '', suggestion)  #将符合模式的字符去除
    if len(suggestion) > 10:
        for k in range(len(suggestion) - 2):
            if suggestion[k : k+2] == '您好':
                reply.loc[i,'完整性'] = reply.loc[i,'完整性'] + 1
                break
        temp = data.留言主题[i]
        n = len(temp)
        for k in range(len(suggestion) - n):
            if suggestion[k : k+n] == temp:
                reply.loc[i,'完整性'] = reply.loc[i,'完整性'] + 1
                break
            elif suggestion[k : k+5] == '留言已收悉':
                reply.loc[i,'完整性'] = reply.loc[i,'完整性'] + 1
                break
        for k in range(len(suggestion) - 4):
            if suggestion[k : k+4] == '答复如下' or suggestion[k : k + 4] == '回复如下
':
                reply.loc[i,'完整性'] = reply.loc[i,'完整性'] + 1
                break
        for k in range(len(suggestion) - 2):
            if suggestion[k : k+2] == '感谢' or suggestion[k : k + 2] == '谢谢':
                reply.loc[i,'完整性'] = reply.loc[i,'完整性'] + 1
                break
        if suggestion[-1] == '日' and suggestion[-3] == '月' or suggestion[-4] == '月'
 or suggestion[-5] == '月':
            reply.loc[i,'完整性'] = reply.loc[i,'完整性'] + 1
        reply.loc[i,'完整性'] = reply.loc[i,'完整性'] / 5

    #可解释性指标构建，构建逗号与句号的权重
    weight1 = 0
    weight2 = 0
    for k in range(len(suggestion)):
        if suggestion[k] == ', ' or suggestion[k] == '; ':
            weight1 = weight1 + 1
        elif suggestion[k] == '。' or suggestion[k] == '! ':
            weight2 = weight2 + 1
    #text 为构建相关性指标中，经过正则处理的留言详情
    number = 0.05 * weight1 + 0.1 * weight2 + len(suggestion)/len(text)
    if number > 5:
        number = 5
    reply.loc[i,'可解释性'] = reply.loc[i,'可解释性'] + number / 5

    #时效性指标构建，计算答复与留言的时间差
    time = math.ceil((data.答复时间[i] - data.留言时间[i]).total_seconds() / 86400)
    if time > 50:
        time = 50
```

```
    reply.loc[i,'时效性'] = reply.loc[i,'时效性'] + 1 - time / 50
    #答复评价
    if sum(reply.iloc[i,:4]) > 3:
        reply.loc[i,'答复评价'] = '完美'
        data.loc[i,'答复评价'] = '完美'
    elif sum(reply.iloc[i,:4]) > 2:
        reply.loc[i,'答复评价'] = '较好'
        data.loc[i,'答复评价'] = '较好'
    elif sum(reply.iloc[i,:4]) > 1:
        reply.loc[i,'答复评价'] = '达标'
        data.loc[i,'答复评价'] = '达标'
    else:
        reply.loc[i,'答复评价'] = '待改善'
        data.loc[i,'答复评价'] = '待改善'

print(reply.可解释性.describe())
print(data.答复评价.value_counts())

data.to_excel('答复评价表.xlsx',index=False)
```