“智慧政务”留言信息的文本挖掘

摘　要

近年来，随着互联网的普及，各种社交软件上的问政平台逐渐成为了成为了我们政府关注民意、体察民意的重要渠道。随着数据量的不断攀升，于是，运用文本分析对留言信息的挖掘具有重大意义。

针对问题一，首先，我们利用 jieba 中文分词工具对留言信息进行分词，并且采用 TD-IDF 算法提取所有留言信息的文本的关键词。然后，我们建立向量空间模型，将每条留言信息生成一个多维向量。利用 K-Means 聚类算法将留言信息聚成七类，每一类分别代表“城乡建设”，“环境保护”，“交通运输”，“教育文体”，“劳动与社会保障”，“商贸旅游”，“卫计卫生”。最后，我们采用 F-Score 评价方法对分类模型进行评价，得出 F_1 为 96.1%，分类结果较好，由此认为我们的模型是合适的。

针对问题二，我们同样利用 jieba 中文分词处理留言信息，提取出文本关键词。我们将特定地点和特定人群的关键词提取出来，同样采用 TF-IDF 算法将文本数据转换为一维向量，再对留言进行分类。然后根据留言点赞数，留言反对数和每句的关键词权重三个指标因子来建立热度评价模型模型，并引用信息量的公式来定义热度计算公式，从而定义合理的热度评价体系。我们对特定地点和特定人群的关键词提取，赋予相应的加权向量之后，通过热度计算公式，我们可以得到每条留言信息的热度指数，再通过分析可得排名前五的热点问题。

针对问题三，我们先对留言主题，留言信息以及答复意见的数据进行清洗与处理。然后，我们用熵权法对答复的相关性，完整性以及可解释性指标因子进行赋值，从求解各指标因子对应的信息熵。利用互信息的相关性公式，我们分析了留言信息与答复的相关性；通过留言信息关键词在答复意见关键词中的比例来分析答复的完整性；通过引用复相关性系数给可解释性下的三个指标因子 (是否引用条例，是否包含相应要点，答复意见是否完整) 赋权来分析答复的可解释性。最终通过答复的三个指标因子分析给出了合理的评价方案。

最后，通过我们对以上问题的数据建模与分析，能给到相关部门一些更好的处理留言信息与留言答复的意见，进而有效的帮助人民群众解决问题。

关键词：文本挖掘，K-Means 聚类，向量空间模型，信息量公式，熵权法

目 录

一、背景与目标 **3**

二、分析方法与过程 **3**

2.1 数据处理 3

2.2 问题一的分析方法与过程 5

2.2.1数据分析 5

2.2.2VSM 向量空间模型 7

2.2.3聚类分析 8

2.3 问题二的分析方法与过程 9

2.3.1数据分析 9

2.3.2热度评价指标模型 11

2.4 问题三的分析方法与过程 13

2.4.1符号描述 13

2.4.2熵权法赋值 13

三、结果分析 **15**

3.1 问题一的结果分析 15

3.1.1聚类结果 15

3.1.2模型分类评价 17

3.2 问题二的结果分析 17

3.3 问题三的结果分析 19

3.3.1相关性 19

3.3.2完整性 20

3.3.3可解释性 23

四、结论 **25**

参考文献 **26**

附录 A 附录 **27**

1.1 问题一代码 27

1.2 问题二代码 29

1.3 问题三代码 34

一、 背景与目标

在当今时代，随着互联网的普及，各大软件上的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。许多群众通过这种渠道向政府反映各类社情民意。因此，相关的文本留言信息量不断增加，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、人工智能等新技术的发展，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。

目标一：根据题目所给的三级标签体系，给留言信息归类，建立一级标签的分类模型。

目标二：用 F-Score 方法评价我们的分类模型。

目标三：针对某一时段内反映的特定地点或特定人群问题的留言信息归类，同时定义合理的热度评价指标体系。

目标四：针对每条留言的答复意见，从相关性、完整性、可解释性等角度对答复意见的质量给出一套合理的评价方案。

二、 分析方法与过程

2.1 数据处理

由于处理的为文本信息，我们先用 python 库中 gensim 将文本信息用向量来表示，在向量维数比较大的情况下，每一个词都可以用元素的分布式权重来表示。因此，我们提取文本信息的特征向量，用来表示留言信息，进而用每个文本的向量聚类。

a. 留言信息的分词处理

在题目给出的文本内容中，在留言详情与答复意见中包含着大量的无关信息，比如有些留言用户对于所反应问题重复表达主观上的认识，以及出现在部分答复意见中的敷衍回答。为了便于对数据进行处理分析，我们需要对文本数据先进行一些基本的清洗处理。

首先是中文分词。常用的分词手段有很多，这里我们采用基于 Python 的 jieba 分词对文本内容进行读取并使用中文 GBK 编码，再调用 jieba 分词得到结果。然而我们发现对于一些人名或者地名 jieba 处理的并不是很好，这里以留言编号为 41458 的留言详情为代表：

你好　　请问　　水　　竹　　湖　　公园　　什么时候　　开工。

因此我们在其中对一些人民地名进行标记处理重新得到结果：

你好　　请问　　水竹湖公园　　什么时候　　开工。

b. 引入停用词

我们通过观察发现上述文本中有很多无效词语比如“看”、“和”以及一些标点符号。这些词语以及标点符号就是停用词。这些停用词的存在会增加对文本数据处理的工作量，影响分析的准确性，因此我们要将它们去除掉。

这里我们通过参数 stop_words 从文件中导入停用词表并将停用词表作用到对应的文本中。这里我们以留言编号为 39444 的留言详情为例：

(1) 原文本：想了解渔民机动船补贴标准，今年只发了 900 元每条船，觉得太少不合理。

(2) 分词处理：想　了解　渔民　机动船　补贴标准　今年　只　发了　900 元　每条船，觉得　太少　不合理

(3) 引用停用词: 了解　渔民　机动船　补贴标准　今年　900 元　每条船　少　不合理

c. 创建文本关键词

为了提高数据的精确性与简洁性，我们接下来要对留言信息做更为细致的处理，这里主要集中在“留言主题”和“留言详情”。在分词处理和引入停用词的基础上，我们利用 python 将留言主题作为 keywords 与留言详情进行匹配，用 jieba 创建自定义词典并将其他无关信息及词语如“谢谢”、“好像”等添加进去进行清洗删除，将与关键词相同的内容以及留言详情中时间、地点、事件以及问句等关键内容形式进行保留。针对“答复意见”考虑到其涉及面的复杂性、广泛性，我们决定保留大部分主题内容，只删除如电话号码、时间日期以及“敬请谅解”、“感谢来信”等相关信息。

总之，我们利用 python 将文本的留言信息进行了分词，并对频率最高的前 100 个关键词进行了频率统计，如下表1：

表 1　关键词词频表

分词	词频	分词	词频	分词	词频	分词	词频
年	10488	小区	2646	教育	1937	社会	1425
市	10396	业主	2641	职工	1863	开发商	1369
月	7613	学生	2641	尊敬	1860	12	1322
县	7015	医院	2625	时间	1816	钱	1312
领导	5450	希望	2552	老百姓	1795	村民	1296
说	4404	请	2471	建设	1795	家长	1291
公司	4068	人员	2386	老师	1791	项目	1276
学校	4026	单位	2384	想	1691	劳动	1269
工作	3901	政策	2308	办理	1666	社保	1246
元	3549	解决	2267	10	1646	费用	1203
政府	3521	居民	2136	管理	1630	更	1200

日	3517	时	2119	未	1627	房屋	1183
部门	3417	教师	2108	退休	1615	投诉	1167
相关	3103	企业	2057	请问	1598	收费	1147
西地省	3000	号	2057	发展	1573	小学	1145
国家	2809	生活	2051	影响	1552	标准	1131
区	2808	孩子	2022	文件	1536	电梯	1130
A	2782	做	2002	人民	1526	工作人员	1123
中	2717	工资	1978	您好	1502	环境	1113
情况	2714	没	1960	教育局	1434	规划	1110

2.2 问题一的分析方法与过程

2.2.1 数据分析

从附件 1 给出的数据中，我们有三级标签的划分体系，然后我们用 excel 画出数据的透视表 (见附件透视表)。从表中，我们知道一级标签下包含的二级标签和三级标签都是交叉的。因此，如果我们考虑先将留言信息归为三级标签，再归为二级标签，最后再归为一级标签的话，那么这样得到的结果很可能会降低查准率，并且模型构建复杂。所以我们不考虑这种分类，我们统一将留言信息直接归结到一级标签。并从附件 2 留言信息的原本的分类中，提取出一级标签，得到图1。

图 1　一级标签示意图

由于我们需要处理的为文本信息，不能直接处理，需要将每段留言信息从文本中提取出来，另存文档。用 python 将每个文档的留言信息转化为句子向量，最后对每个文档进行相似性分析，得到每段信息的分类。

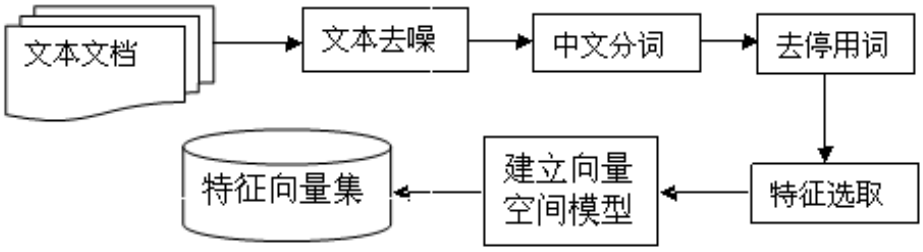


图 2　问题一流程图

经过我们的数据处理得出来的分词，从这些频率最高的分词中，我们也可以从中定义关键词，但由于这些词还存在了一些地点名词、年份、数量词等一些名词，这些词对于问题一的留言信息分类并没有太大作用，我们将会对这些词做上标记并将词去除，进而重新定义关键词。由此我们生成关键词词频的频率直方图和词云图，如图3和图4：

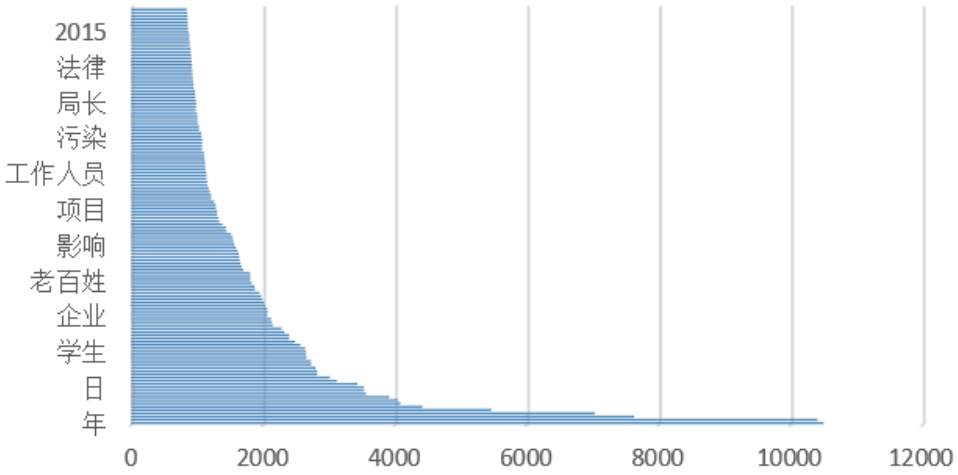


图 3　词频直方图

图 4　关键词的词云图

图3为去除掉无意义名词后的高频率名词的频率直方图，我们可以将这些关键词与我们的一级分类标签进行相似度分析，先将关键词进行归类，进而再对文本信息归类。图4是对频率出现最高的分词生成的词云，从图上我们可以清楚直观的浏览到所出现的分词，也可以从图中词的大小知道各个分词的频率。

2.2.2 VSM 向量空间模型

我们从留言信息中随机取出 20% 的留言做为训练集，80% 的留言做为测试集，为了将文本信息转为向量，我们建立了基于留言信息的向量空间模型。在的向量空间模型中，一个文档被表示称一个由从文档中抽取的关键词及其权重组成的向量，权重表示了关键词在文档中和在整个文档集中的重要性。

一个特征项在一个文档向量中的权重可以由多种方式来决定。我们用 TF-IDF 的方法来决定特征项在文档中权重，该方法中一个特征项的权重取决于两个因素：特征项 j 在文档 i 中出现的频率 $tf_{i,j}$ 和它在整个文档集中出现的频率 df_j。准确地说，特征项 j 在文档 i 中的权重为：

$$w_{i,j} = tf_{i,j} \times idf_j = tf_{i,j} \times \frac{\log N}{df_j}$$

其中 N 是文档集中文档的总数，idf 表示逆文档频率。这种方法在一个小的文档集合中给出现频率高的特征项分配高的权重。

2.2.3 聚类分析

a. K-Means 聚类算法

K-Means 算法是一种基于划分方法的经典聚类算法，该算法始于一个簇的中心集合，该集合是随机选择的或者根据一些启发式方法选择。在每次的迭代过程中，每个样本点根据计算相似度被分配到最近的簇中。然后，重新计算簇的中心，也就是每个簇中所有数据的平均值。每个簇的中心就是所有这个簇的所有样本点的中心：

$$\mu_{\mathrm{k}} = \frac{1}{N_k}\sum_{q=1}^{N_k} x_q$$

其中是属于簇 k 的样本数目，是指簇 k 的中心。

K-Means 算法步骤：

输入:包含 N 个数据对象的数据集合。

输出:K 个聚类簇的集合。（K 为一级标签的个数）

Step1: 从我们的数据文本向量中任意选择 K 个数据对象作为初始聚类中心；

Step2: 根据数据对象与 K 个簇的相似度，将每个对象再分派到最相似的簇中；

Step3: 计算每个簇中所有对象的均值形成新的簇中心；

Step4: 重复 (2)(3) 步骤，直到聚类结果不再改变

Step5: 输出结果

b. 算法流程图

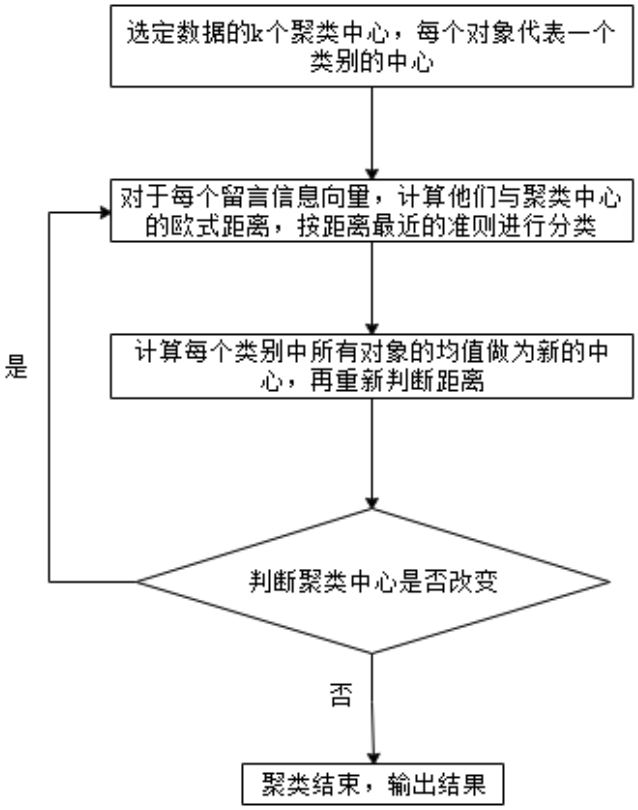


图 5　K-Means 算法流程图

2.3 问题二的分析方法与过程

2.3.1 数据分析

问题二主要是以 jieba 库的中文分词为初始基础，对附件三中留言主题一系列数据进行分析，前期经过对留言数据的初步清洗，通过统计每条留言的各分词数量以及相应的词频词性，可以得出一部分的特定地点。类似于问题一的分析方法，通过去停用词以及 jieba 分词后，采用 TF-IDF 算法，对每个分词进行由文本数据向权重向量的转变，之后通过以上找到的相关数据，根据各分句的权重加权排序可将问题大致进行分类。另一方面，从热度的信息论角度出发为跳板，将热度与信息论中的自信息量结合，借用信息量的计算公式，引出热度指标计算公式，同时将留言点赞数、反对数以及关键词的权重作为热度评价指标，由热度指标公式也算出了各留言的相应热度，通过结合之前的留言归类，可进一步加权得出归类的留言热度排名，之后就可得出将前五的热度问题。

我们在对数据进行简单的去停用词和分词操作后，采用 TF-IDF 算法用以评估一个字或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。由上述操作后，

我们得到问题二的流程图，如图6所示。

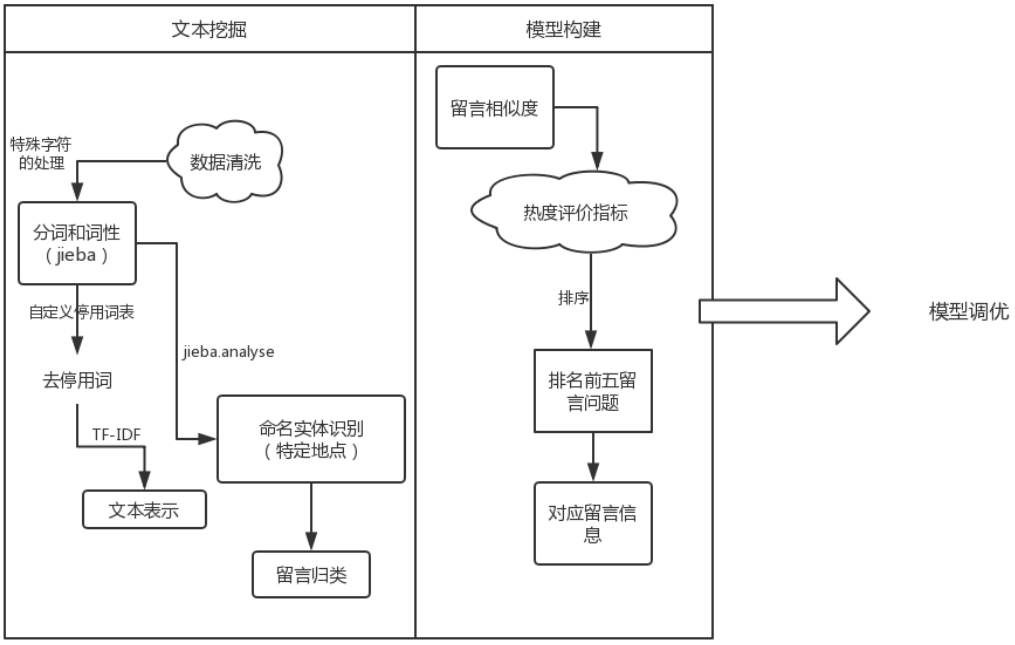


图 6　问题二的流程图

在题给出的附件三中，存在许多用户留言信息，题目要求对用户留言进行特定地点或人群分类，因此考虑用 python 进行附件 3 表格“留言主题”一列的导入，并将提取出的数据经过代码转换后输出为一个“result.txt”文本，再对此文本进行数据导入，导入 python 其中的 re 模块，利用其中的正则表达式，对部分特殊字符进行预处理。

结合之前进行的词频统计，基于所给的实时数据，我们可以按照每句关键词的相似度，以此大致将附件三中的用户留言问题进行初步归类。

表 2　关键词的词频和词性

词	计数	词性	词	计数	词性
小区	548	n	西地省	159	ns
扰民	278	n	噪音	147	n
街道	199	n	施工	140	vn
咨询	171	vn	社区	132	n
建议	167	n	国际	127	n

类似于问题一的方法，去停用词，进而对留言数据进行进一步的文本筛。然后去除常见的与文本表示无关的部分词语，将最终处理好的留言数据保存为“final.txt”文件，方便之后的关键词提取并计算权重。将词语进行好清洗、停用和分词后，我们采用 TF-IDF 算法将文本转换为一维的数字向量。

我们知道 TF-IDF 值越大，则这个词成为一个关键词的概率就越大，某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。如此在 python 中运用 TF−IDF 算法，通过代码的调整，我们将 final 输出中的每一条留言分别进行关键词的提取以及权重的计算，将关键词信息转为权重向量并进行降序排列的输出，部分结果整理如下：

```
Prefix dict has been built successfully.
 米阳婚纱艺术摄影合法纳税
米阳 2.39095350058
艺术摄影 2.39095350058
婚纱 1.9977309293099998
纳税 1.6503406827620002
 咨询道路命名规划步成果公示城乡门牌题
门牌 1.8654379084833332
城乡 1.2388298209783333
命名 1.155147830345
成果 1.1188367479766665
 映春华镇金鼎村水泥路水户题
水泥路 4.171461096966667
水户 4.097079913166667
```

图 7　部分留言主题下的关键词及权重

2.3.2 热度评价指标模型

考虑到如今微博、微信等网络问政平台的大众化，用户之间具有关注和被关注的关系，留言本身由点赞和反对的关系。题目中热度，从逻辑上来说，指的是人们的观点、话题或者是某一次受关注的程度，从信息量的角度解释是人们留言中所包含的信息量。因此，我们引入每条留言的点赞数、反对数以及关键词的权重，几个最具代表性的留言元数据来作为表征每条留言热度的参数，即热度的评价指标。

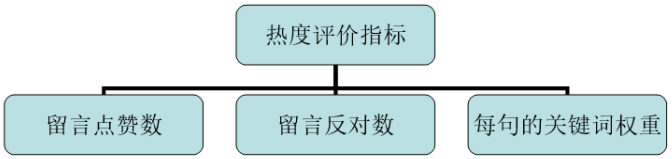


图 8　热点问题评价指标

而对于热度计算方式的引用思考，来源与上述信息论中信息量的描述：一个事件信息量的大小与该事件发生的概率有关。极少发生的事件一旦发生是容易引起人们关注的，而司空见惯的事不会引起注意，也就是说，极少见的事件所带来的信息量多。如果用统计学的术语来描述，就是出现概率小的事件信息量多。因此，事件出现得概率越小，信息量愈大即信息量的多少是与事件发生频繁（即概率大小）成反比，而此处的信息量就可以当作本题的热度。

如有 a 件事情，当事件 X_i 发生，则表示 X_i 所含有或所提供的信息量为

$$H(X_i) = -\log_a p_i$$

类比于信息量的计算，假设留言 m 点赞数为 d、反对数为 f 以及关键词的权重 qz，则此留言的热度计算公式定义为

$$H(m_i) = -\log_a \frac{1}{\mathrm{d} + \mathrm{f} + \mathrm{qz}}$$

2.4 问题三的分析方法与过程

2.4.1 符号描述

符号	定义
S	答复意见的质量在评价方案中的分数；
D	答复的相关性；
D_i	答复相关性的各项指标；
D_{ij}	各项答复相关性指标的值；
a_1	答复的相关性在答复意见质量评价方案中的权重；
C	答复的完整性；
C_i	答复完整性的各项指标；
C_{ij}	各项答复完整性指标的值
a_2	答复的完整性在答复意见质量评价方案中的权重；
I	答复的可解释性；
I_i	答复可解释性的各项指标；
I_{ij}	各项答复可解释性指标的值；
a_3	答复的可解释性在答复意见质量评价方案中的权重；

其中 $\mathbf{D}=[\mathbf{D_1},\mathbf{D_2},...,\mathbf{D_n}]'$，$\mathbf{D_i}=[\mathbf{D_{11}},\mathbf{D_{12}},...,\mathbf{D_{1j}},...,\mathbf{D_{1m}}]'$。$C$ 及 I 同理。基于以上变量我们建立线性规划模型 $S=a_1D+a_2C+a_3I$。

2.4.2 熵权法赋值

附录中针对问题三的附件 4 为我们提供了有关留言主题、留言详情、答复意见等文本数据。我们首先要进行文本数据的预处理。附件四是一个庞大的文本数据，而其中存在大量冗余的价值很低甚至没有价值的数据，如果保留这些留言数据会对分析造成很大的影响得到的结果也必然存在问题。因此我们需要对数据进行相应的清理、筛选、对照等处理以保障数据的规范性。在这里，我们使用熵权法对各个指标因子赋值。

a. 数据标准化

由于使用了不同单位的指标，为了便于计算分析我们需要对不同指标的数据进行标准化处理并将全部指标压缩至 [0□1] 之间。下面我们给出标准化的公式：

$$Y_{ij} = \frac{D_{ij} - \min(D_i)}{\max(D_i) - \min(D_i)}$$

b. 求解对应的信息熵

在这里，结合相关的理论依据以及附件 4 提供的的数据，我们决定引入信息熵的概念。信息熵是信息论中用于衡量信息一个重要的指标，是反应了衡量信息的不确定性以及混乱程度的指标。根据信息论中信息熵的定义结合文本数据的各项指标，我们通过简单的简化处理得到信息熵的计算公式：

$$E_j = -\ln(\frac{1}{n}) \cdot \sum_{i=1}^{n} p_{ij} \cdot \ln(p_{ij})$$

其中 $p_{ij} = \frac{Y_{ij}}{\sum_{i=1}^{n} Y_{ij}}$。

c. 确定权重

在确定信息熵的基础上，我们计算出各个指标的信息熵 E_1,E_2,……,E_3。通过各个指标的信息熵我们得出答复相关性在答复意见质量的评价方案中所占的权重为：

$$a_1 = \frac{\sum_{i=1}^{n} Q_i}{n}$$

其中 $Q_i = \frac{(1-E_i)}{n-\sum_{i=1}^{n} E_i}$。

这样通过上述的熵权法我们得到了答复的相关性、完整性、可解释性在答复意见评价系统的所占的权重。

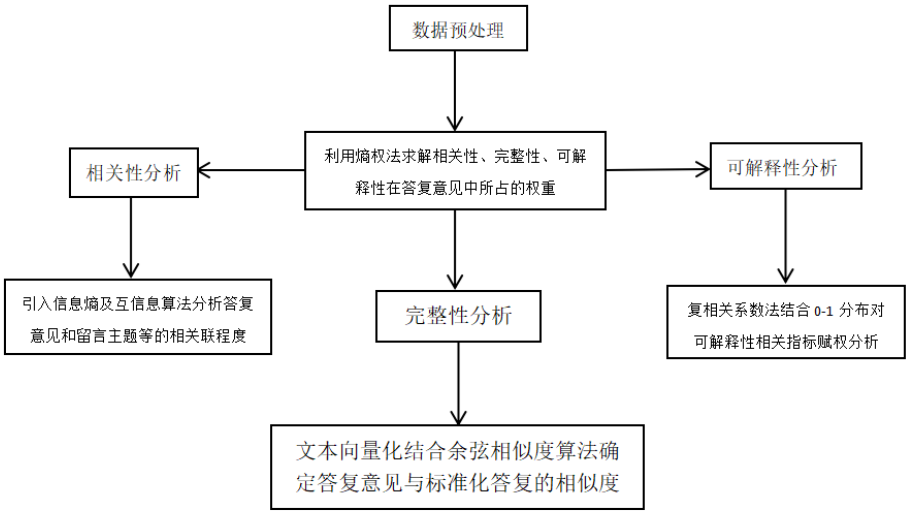


图 9　问题三的流程图

三、 结果分析

3.1 问题一的结果分析

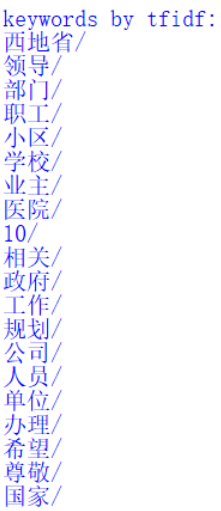
3.1.1 聚类结果

a.在进行聚类挖掘之前，首先要对文本文档进行数据预处理，其中主要包括以下几步：

(1) 中文文本分词，结果见附件“分词.txt”。

(2) 采用停用词列表去停用词。

(3) 用 python 对文本文档集进行特征选取，这里采用 tf-idf 算法和 textrank 算法提取关键词，选取的特征项如图11和图12所示。



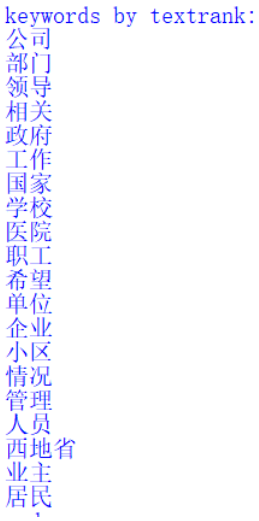


图 11　TF-IDF 算法得出的关键词　　**图 12　textrank 算法得出的关键词**

从结果中，我们看到 tf-idf 算法提取出的特征项中存在数量词“10”，而这类文本信息对我们分类是无意义的，对比两类算法，显然 textrank 算法提取出的关键词更佳。

表3表示，用 python 训练后的语料库，再输入“城乡建设”，“环境保护”，“交通运输”，“教育文体”四个一级标签后，找到与其相似性最大的前 20 个词语。所以我们可根据与一级标签有 85% 以上的相似度的词语来创建关键词，从而进行分类。当然，为了提高精确性，也可以适当提高关键词的相似度标准。

表 3　分词与一级标签的相似度

城乡建设		环境保护		交通运输		教育文体	
属下	0.9303	专项	0.9721	文物保护	0.9558	人才	0.9223
送交	0.9261	评价	0.9575	十大	0.9541	基础教育	0.8886
主导	0.9091	消防法	0.9563	产权制度	0.9476	力量	0.8825
审批同意	0.9074	报告书	0.9467	争创	0.9450	群众路线	0.8785
同酬	0.8877	论证会	0.9429	广播电视	0.9435	绵薄之力	0.8772
遗留问题	0.8867	环境影响	0.9381	叶绿	0.9416	咨询服务	0.8721
牵头	0.8850	突发事件	0.9362	市级	0.9382	阳光	0.8715
两级	0.8847	国家有关	0.9355	新闻出版局	0.9359	高校	0.8572
研究	0.8786	规范	0.9345	综合执法	0.9356	传递	0.8567
城乡规划	0.8781	许可	0.9185	信息化	0.9346	实践	0.8548
执法机关	0.8780	目的	0.9175	牵头	0.9341	民办学校	0.8491
批准	0.8750	通信	0.9171	智慧	0.9337	职高	0.8477
下属	0.8673	主体	0.9147	最先	0.9300	体现	0.8452
住建	0.8635	建设项目	0.9142	建言	0.9275	义务教育	0.8438
龙头企业	0.8627	秉公	0.9123	局在	0.9268	搞些	0.8423
委员会	0.8624	采取有效	0.9116	州市	0.9260	公平	0.8423
添砖加瓦	0.8595	第十五条	0.9094	交警大队	0.9217	公办	0.8392
文物保护	0.8557	矿产资源	0.9090	执法局	0.9216	践行	0.8381
招入	0.8541	招投标	0.9079	欢迎您	0.9204	师资	0.8378
属	0.8539	完善	0.9053	业务管理	0.9196	民办	0.8365

b. 对于问题 1，我们需要对分类标签与特征项进行聚类，进行距离判别。所以我们用 python 编程，对附件 2 中 9210 条留言信息进行分类，得到结果如表4。从表中我们可以得到 7 个类中心分别属于“城乡建设”，“环境保护”，“交通运输”，“教育文体”，“劳动与社会保障”，“商贸旅游”和“卫生卫计”。

表 4　分类表

聚类中心	类中心 1	类中心 2	类中心 3	类中心 4	类中心 5	类中心 5	类中心 7
a. 城乡建设	1852	157	0	0	0	0	0
b. 环境保护	0	855	51	0	0	0	32
c. 交通运输	0	13	582	0	0	18	0
d. 教育文体	0	75	0	1442	72	0	0
e. 劳动与保障	125	0	0	0	1694	150	0
f. 商贸旅游	0	81	0	0	0	1134	0
g. 卫生卫计	21	0	0	0	31	0	825
所属类别	a	b	c	d	e	f	g

3.1.2 模型分类评价

根据题目信息，我们需要用 F-Score 评价方法对我们的分类模型进行评价。由附件 2 原本分好类的一级标签与我们得出来得一级标签做对比，通过 excel 数据处理，得到查准率与查全率。

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

其中 P_i 为查准率，R_i 为查全率，n 为分类标签数。

同时通过表4，我们先计算各个标签的查准率，然后得出 7 个一级标签的平均预测精度为 92.51%，由于我们的所有的留言都归类，所以我们的查全率为 100%，分类效果中等偏上。再通过 F-Score 评价方法，我们得出 F_1 为 96.1%，结果较好，所以我们的模型是合适的。

3.2 问题二的结果分析

我们从之前的命名实体识别操作中，得出了部分特殊地点及其词性，因此在代码里统一对每一句的留言提取四个关于地点的关键词并得出相应的权重，进行对应向量加权后，我们将权向量赋值返回原文件即附件 3 中，根据加权出的 TF-IDF 向量的总和排序，部分结果如下图12所示。可知，由加权权重进行归类效果较好。

	A	B	C	D	E	F	G	H	I	J	K	L
1	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		权重			
2	208285	A909205	投诉小区附近搅拌站噪音扰民	2019-12-15 12:32:11	门窗还是有很大噪音，吵得	0	24		39.85878			
3	255008	A909208	投诉小区附近搅拌站噪音扰民	2019-11-18 12:23:22	地方搬过来的，体会最深的就	0	0		39.85878			
4	261072	A909207	投诉小区附近搅拌站噪音扰民	2019-11-23 23:12:22	，在居住区建立搅拌站水泥	2	9		39.85878			
5	266665	A00096279	投诉小区附近搅拌站噪音扰民	2019-12-04 17:23:22	居民的正常生活。想问政府	0	0		39.85878			
6	201447	A00020543	国道107距我家仅3米，相关政府部门为何不同意	2019/7/22 17:04:08	完全统计，整栋房屋开裂约	0	2		33.53058			
7	229903	A00020543	国道107距我家仅3米，相关政府部门为何不同意	2019/7/22 17:05:04	完全统计，整栋房屋开裂约	0	0		33.53058			
8	273925	A00020543	国道107距我家仅3米，相关政府部门为何不同意	2019/7/18 10:47:31	完全统计，整栋房屋开裂约	0	3		33.53058			
9	278907	A00020543	国道107距我家仅3米，相关政府部门为何不同意	2019/7/18 10:48:28	完全统计，整栋房屋开裂约	0	0		33.53058			
10	239737	A000107463	A市能否设立南塘城轨公交站？	2019/10/31 21:19:59	，南塘小学，A市一中城南	0	0		33.08753			
11	264088	A000107463	A市能否设立南塘城轨公交站？	2019/10/31 21:17:22	，南塘小学，A市一中城南	0	1		33.08753			
12	343985	A108051	A市能否设立南塘城轨公交站？	2019-10-31 21:19:59	，南塘小学，A市一中城南	0	0		33.08753			
13	190812	A00095451	A市江山帝景新房有严重安全隐患	2019/5/30 17:34:02	雨雪天气后过道全部是水和	0	0		30.18316			
14	289893	A00095451	A市江山帝景新房有严重安全隐患	2019/5/30 17:20:53	雨雪天气后过道全部是水和	0	0		30.18316			
15	319659	A023956	A市江山帝景新房有严重安全隐患	2019-05-30 17:34:02	雨雪天气后过道全部是水和	0	0		30.18316			
16	189950	A909204	投诉A2区丽发新城附近建搅拌站噪音扰民	2019-11-13 11:20:21	的地方建搅拌站。可想而知，	0	0		29.89408			
17	231136	A909204	投诉A2区丽发新城附近建搅拌站噪音扰民	2019-12-02 11:20:21	告。距离上次投诉已经过去	0	0		29.89408			
18	253040	A909202	投诉A2区丽发新城附近建搅拌站噪音扰民	2019-12-04 12:10:21	家里根本无法正常休息！投	0	0		29.89408			
19	237116	A00031618	请A市加快一圈二场三道建设力度	2019/2/3 14:15:28	马路中间穿行导致交通事	0	2		27.03693			
20	240662	A00031618	请加快A市一圈二场三道建设力度	2019/4/18 16:38:15	众多，连过街设施都没有，	0	2		27.03693			
21	275514	A00031618	请A市加快一圈二场三道建设力度	2019/4/22 17:29:22	一个过街天桥都没有，导	0	2		27.03693			
22	217380	A00093113	郡未来实验学校针对四年级学生上调学费是否违	2019/5/30 10:55:02	一二三年级的学生，适用之	0	3		26.5076			
23	274341	A00021619	楚郡未来实验学校针对四年级学生上调学费是否	2019/5/31 16:23:42	的。三年级所有家长认为：	0	1		26.5076			
24	289606	A00093113	楚郡未来实验学校针对四年级学生上调学费是否违	2019/7/16 22:49:58	部门能够明确告知我们家长	0	0		26.5076			
25	194022	A00042107	坚决反对在A7县诺亚山林小区门口设置医院	2019/7/8 10:39:54	受到干扰。本小区的业主是	0	1		24.15741			
26	210107	A00042107	坚决反对在A7县诺亚山林小区门口设置医院	2019/7/8 10:38:38	受到干扰。本小区的业主是	0	14		24.15741			
27	226871	A000726	坚决反对在A7县诺亚山林小区门口设置医院	2019/8/16 8:37:43	境优良的家里，实际却是拍	0	1		24.15741			
28	252413	A000726	坚决反对在A7县诺亚山林小区门口设置医院	2019/8/16 8:36:38	气中充斥着无数的病菌。人	0	0		24.15741			

图 12　热点问题分类结果图

由热度评价指标模型，我们再次在 python 中计算每句留言的关键词权重 (之前由于按特定地点抽取地名类名词，有所偏差)，得到部分结果如下图13：

举报食家饭店违法经营破坏环境
食家 1.7078239289857142
饭店 1.1162718464371428
举报 1.0696026706871429
违法 1.0157972023514286
现原高塘岭镇范围轮电动车遍开花
现原 2.39095350058
高塘岭 2.39095350058
开花 1.61200719892
电动车 1.51135944357
洞井街道欧莱雅郡期复式顶违建题愈发严重
洞井 1.4943459378625

图 13　建立模型后的关键词指标权重

同样，将各关键词导入对应留言表格中进行权重加权操作，最后可得到每条留言的热度指数如表5所示。

表 5　部分留言的热度指数

留言编号	留言主题	热度指数
213584	投诉 A 市伊景园滨河苑定向限价商品房违规涨价	0.31868398
244342	投诉 A 市伊景园滨河苑定向限价商品房违规涨价	0.20858616
275083	反映 A3 区保利西海岸小区幼儿园的问题	0.37331157
277350	反映 A3 区保利西海岸幼儿园的一些问题	0.25341523
212601	A 市 A5 区利聚网平台不让提现	0.30966998
237504	A 市 A5 区利聚网平台不让提现	0.32590526

完成上述的留言归类以及热度评价操作之后，我们能够进一步得到每一相似部分留言问题的总热度指标，由此我们可以按照归类排序，按题目所给的表一格式给出热度排名前五的热点问题，保存为相应文件“热点问题表.xls”，再按题目表二格式给出相应热点问题对应的留言信息，并保存文件“热点问题留言明细表.xls”。

3.3 问题三的结果分析

3.3.1 相关性

答复的相关性，顾名思义即答复意见的内容是否与问题 (也就是留言主题以及留言详情) 有关。这里我们将答复意见质量评价方案中答复的相关性主要理解为问题关键词与答复的匹配程度，其中包括关键词是否匹配、关键词匹配程度的高低。

a.关键词匹配度

我们主要利用 python 软件来确定关键词匹配度。首先我们利用 jieba 对文本中的“留言主题”进行分词、去停用词以及提取关键词等操作并创建词典将提取内容设置为第一组关键词，再简易使用 TF-IDF 算法分别将“留言详情”以及“答复意见”中涉及到时间、地点、人物、时间的关键词提取出来并设置为第二组关键词、第三组关键词。我们随机抽取了一条留言作为例子进行说明。

(1) 留言编号：2555

(2) 留言主题关键字：加快　提高　民营幼儿园　老师　待遇

(3) 留言详情关键字：民营幼儿园　众多　教师　超负荷工作　收入低　大力倡导　加大　工作压力　雪上加霜　加快　提高　民办幼儿园　待遇　降低

(4) 答复意见关键词：改善　提高　民办幼儿园　待遇　签订合同

工资收入　　保障　　加强　　明确　　教职工　　合理　　社会保障　　推进　　公办幼儿院　　依法

b.关键词匹配度处理

首先我们在 python 中利用 TFIDF 算法以及词袋构建引入 jieba、pandas、numpy 等将近义词语进行分类，比如将意思相近的“加快，改善，提高、加强”等词语分类在一起。然后我们运用 0-1 分布将前两组关键词与第三组关键词进行匹配，匹配成功输出结果为“1”，匹配失败输出结果为“0”，观察并统计匹配成功的关键词的种类数目。接下来我们引入 Python 中的 collections() 函数统计输出结果为“1”即相应匹配的各个关键词在第三组关键词出现的次数，并计算次数之和。最后通过简单计算分别得出这些关键词在“答复意见”关键词以及答复意见内容中所占的比例并记为 p_i 与 m_i。

c.相关性分析

通过对文本数据的分析和相关资料的查询，我们采用 ID3 算法来评估“答复意见”与“留言主题”、“留言详情”之间的相关性。ID3 算法的核心是用来衡量一元模型 (单一事件) 中信息不确定性以及相关性的信息熵、条件熵与“互信息”。

互信息是衡量信息之间相关性的指标。当两个信息完全相关时互信息为 1，不相关时为 0。信息熵和条件熵都是衡量信息不确定性与相关性的指标，信息之间的的关联性越强，信息熵的值越大，条件熵的值越小，而影响熵值的主要因素是概率。这里我们用第一、二组关键词在“答复意见”关键词中出现的频率 p_i 近似概率并得到相应的模型。我们可以得到信息熵的相关性公式：

$$E(T) = \sum_{i=1}^{c} -p_i \cdot \log_n(p_i)$$

条件熵得相关性公式：

$$M(T) = \sum_{i=1}^{c} p_i \cdot m_i$$

互信息的相关性公式：

$$G(T) = E(T) - M(T)$$

其中 T 为每一个留言所反应的事件，p_i 为第一二组关键词在“答复意见”关键词中所占的比例，m_i 为第一二组关键词在“答复意见”内容中所占的比例，c 为匹配成功的关键词的种类数目，n 为匹配成功的关键词在第三组关键词中出现次数之和

3.3.2 完整性

通过相关的查阅文献我们最终将“答复的完整性”理解为“是否满足某种规范”，这里我们引入相似度的概念作为一般答复与规范化答复比较的指标。因此我们需要解决两方面的问题：一是如何制定一个规范化的答复标准作为对照，便于其他的答复进

行比较以确定其是否满足标准；二是找到合适的比较方法即合理的算法、模型对相似度进行解读、计算使其成为能够量化的指标便于比较。这里我们分别进行讨论。

a.规范化答复的确定

这里我们将之前相关性分析里关键词匹配度处理的结果作为基础，绘制出“留言主题”、“留言详情”关键词 (记为 A,B 组关键词) 分别在“答复意见”关键词 (记为 C 组关键词) 以及答复意见内容 (简记为答复) 中所占的比例图像，如图14和图15所示。

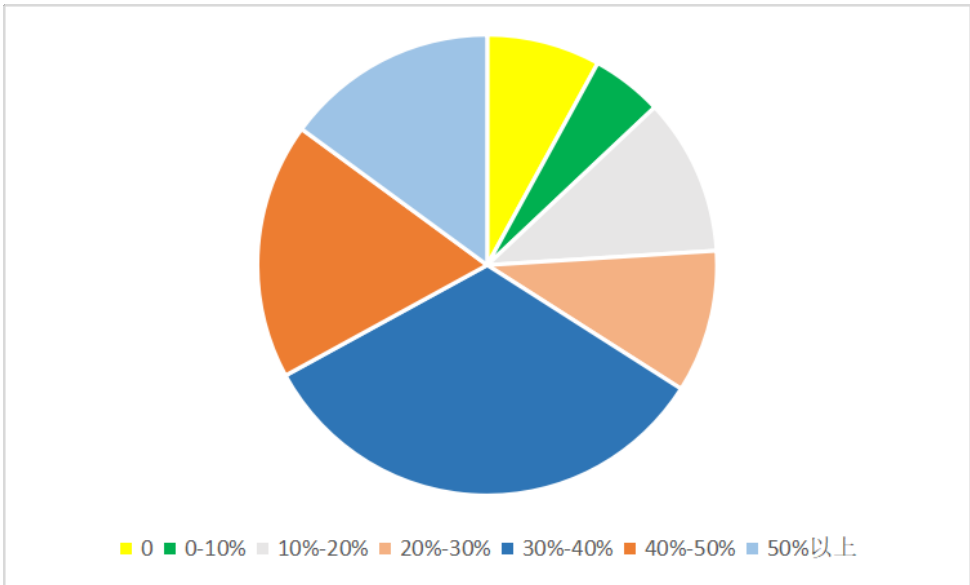


图 14　A，B 组中的关键词在 C 组中的比例分布

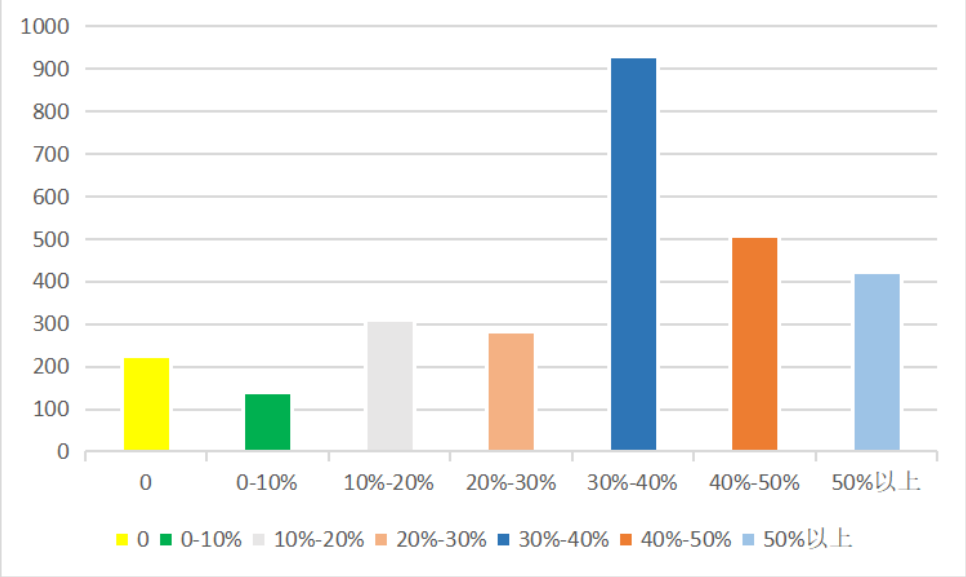


图 15　A，B 组中的关键词在答复中所占比例

通过图表观察，我们将 A,B 组关键词在 C 组中比例分布在 20%~50% 的部分以及 A,B 组关键词在答复中比例在 10%~40% 的部分作为主体，以数据预处理中的创建文本

关键词的操作为基础，删除相关关键词保留答复中的其他部分，我们就得到了许多答复的无关内容。我们将这些内容主要分成问候语、问题解答以及总结语三个部分，考虑到问题解答部分的复杂性、专业性，这里不做具体的考虑。因此我们确定了一个完整的规范化答复意见的内容分布，即问候语、问题解答和总结语。

b.比较方法的确定

在已经确定了规范性答复的情况下，我们需要运用合理的模型和算法将一般答复与规范化答复进行比较来确定其是否满足这种规范。

这里我们采用余弦相似度算法来计算、比较文本之间的相似度。对应的余弦值越大，相似度越大，文本之间的“距离”越小；余弦值越小，相似度越小，文本之间的“距离”也越大。在这之前我们已经对一般答复意见以及规范性答复意见进行了相应的 jieba 分词处理以及关键词分类，这里我们直接进行分词编码、词频向量化等处理并套用余弦函数计算两个句子的相似度。

首先我们使用词袋 (BOW) 模型在忽略它们单词顺序以及语法等要素的前提下，将两个文本 A，B 看做是若干个词汇的集合并假定每个单词的出现都是独立的，把两个文本中出现的分词分别构造得到两个列表 listA，listB，进而把两个列表 listA，listB 放在同一个集合 set 并转换为字典 dict，其中 key 为 dict 中出现的词语，value 为集合 set 中必须出现的位置。

第二步我们将 listA，listB 进行编码，把每个词语转换为在集合 set 中出现的位置得到 listAcode 与 listBcode，其相应的一行 n 列矩阵分别简记为 A_1 与 B_1。

第三步将 listAcode 与 listBcode 进行 oneHot 编码，即计算每个分词出现的次数并在相应的位置上将词语转换为次数得到 listAcodeoneHot 和 listBcodeoneHot，其相应的 1 行 n 列的矩阵简记为 A_2、B_2。

最终引入余弦相似度算法得到公式：

$$similarity = \cos\theta = \frac{A_1A_2' + B_1B_2'}{\sqrt{(A_1A_2')^2 + (B_1B_2')^2}}$$

这里我们以留言编号为 35818 的答复意见以及规范性答复中的问候语为例进行说明。

(1) 编号 35818：网友　　你好　　信件　　收悉　　核查　　相关　　信息　　答复　　如下

(2) 规范性答复：尊敬　　网友　　您好　　反应　　内容　　收悉　　调查　　回复　　如下

通过以上算法及模型我们得出 $cos\theta = 0.8427$。

3.3.3 可解释性

广义上的可解释性是指我们在需要了解或解决一件事情的时候我们可以获得我们需要的、足够的、可以理解的信息。在这里我们将答复意见的可解释性与答复意见内容中的相关解释联系在一起。一般来说，我们认为一个答复意见的可解释性与其是否引用条例、是否包含相应的要点以及答复的详细程度有关。这里我们引入复相关系数法结合 0-1 分布 (即若为“是”输出结果为“1”，若为“否”输出结果为“0”) 对以上三个指标进行赋权。

a. 求出各指标的相关系数矩阵

我们使用三个指标，分别为“是否引用条例”、“是否包含相应的要点”、“答复意见是否完整”，并将三个指标记为 X_1，X_2，X_3。其中的“答复意见是否完整”与之前的完整性分析类似，可以直接引用。再通过对关键词的筛选确定“是否引用条例”以及“是否包含要点”的个数，于是我们就得到了一个两行三列的相关系数矩阵:

$$R = (r_{11}, r_{12}, r_{13}; r_{21}, r_{22}, r_{23})$$

其中 r_{ij} 为第 i 个指标的情况。

于是我们对答复意见内容反复提取相应关键词得到如图16分布情况。

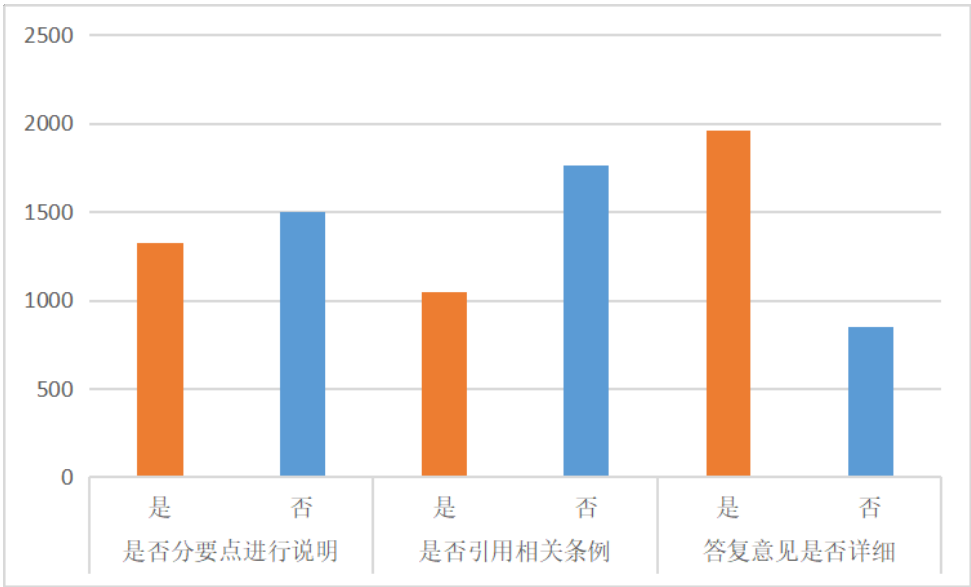


图 16　可解释性基本指标分布

b. 计算指标之间的复相关系数

运用相应算法计算某一个指标与另外两个指标之间的复相关系数。这里我们对 R 进行一定的分解处理。这里假设是对第 m 个 $(m = 1, 2, 3)$ 指标分析，求解其与其他两个指标的相关性系数。对 R 进行处理得到得到

$$R' = \begin{pmatrix} R_{m-1} & r_m \\ r_m{}' & 1 \end{pmatrix}$$

进而得到 X_m 对其他指标的复相关系数:

$$p_m = r_m{}' \cdot R_{m-1} \cdot r_m$$

其中 R_{m-1} 为其他两个指标得相关系数矩阵，$r_m = (r_{1m}, r_{2m})'$。

c.求解各指标权数

对复相关系数求倒数并进行归一化处理就得到了各指标权数，得到：

$$W_j = \frac{1/p_j}{\sum_{j=1}^{3} p_j}$$

通过以上方法结合运算我们最终达到 $W_2 = 0.4863$，$W_2 = 0.4159$，$W_3 = 0.0957$。这也符合相应的主观预期，当某种指标与其他指标重复的信息越多，在综合评价中的作用就越小，就应该赋予较小的权重。这里“答复意见是否详细”与完整性分析中重复的部分比较大，因此在可解释性分析中所占权重较小。

表 6　随机抽取留言的各指标值

留言编号	互信息熵值	余弦相似度值	复相关系数值
5328	0.3491	0.8923	0.9371
5607	0.2983	0.8536	0.9123
5948	0.1375	0.7296	0.8021
123860	0.0059	0.0007	0.0004
33375	0.2634	0.8355	0.9085
35455	0.1922	0.7601	0.9264
75000	0.0000	0.0000	0.0000
82255	0.1125	0.7192	0.8981

在这里我们随机从附件四抽取了 8 条留言，通过对三组数值的计算结果进行对比以及主观上对留言回复内容的比较判断，以留言编号 5948 与 35455 为例，我们主观上认为留言 5948 的回复内容相较于留言 35455 更清晰准确，更贴合实际需求，也更容易理解。但通过数值比较我们发现在可解释性对应的数值即复相关系数值上，留言 5948 在可接受范围内低一些。

四、 结论

建立基于自然语言处理的智慧政务系统可以有效的对各网络平台社情民意留言信息的数据分析，进而有效的帮助政府了解民意，更好的服务于我们的群。本文通过K-Means 聚类算法，结合三级标签的划分体系，有效的将我们的留言问题归为七类。我们的政府可以分别从这七类问题出发，解决民众在生活中遇到的各类问题。

与此同时，对热点问题的数据挖掘能指导政府更好的帮助到群众，从而制定相应的解决政策。本文利用信息量的计算公式建立了合理的热度评价体系，能帮助政府从大量的文本信息中提取出热点问题和及时了解社会热点信息，进而更好的帮助到群众。

然而，对每条留言的答复意见对有效解决群众问题也至关重要，政府必须针对每条留言的相关问题给出相应答复，才能有效帮助到群众。本文从答复的相关性、完整性、可解释性三个角度出发，制定了一套合理的答复意见评价方案，用以帮助相关部门更好的给出相应问题的回答。

参考文献

[1] 姜启源; 谢金星; 叶俊. 数学模型 [M]. 北京: 高教出版社，2003.8.

[2] 孙吉贵; 刘杰; 赵连宇. 聚类算法研究 [J]. 软件学报, 2008, 19(1): 48-61.

[3] 范国凤; 刘璟; 姚绍文; 栾桂凯. 基于语义依存分析的图网络文本分类模型 [J]. 计算机应用研究，2020，v.37，1-5.

[4] 唐晓波; 向坤. 基于 LDA 模型和微博热度的热点挖掘 [J]. 图书情报工作,2014,v.58;No.498,60-65.

[5] 杨小兵. 聚类分析中若干关键技术的研究 [D][J]. 博士, 浙江大学, 2005.

[6] 杨宇. 多指标综合评价中赋权方法评析 [J]. 理论新探.2006：17-19.

[7] 陈琼华. 综合评价中的赋权方法 [J]. 知识丛林.2013：118-119.

[8] 郭理; 张恒旭; 王嘉岐; 秦怀斌. 基于 Trie 树的词语左右熵和互信息新词发现算法 [J]. 现代电子技术.2020：65-69.

[9] 百度百科.https://baike.baidu.com/item/AHP/1672?fr=aladdin, 2020/3/28.

[10] MBA 智库百科.https://wiki.mbalib.com/wiki/AHP, 2020/3/28.

附录 A　附录

1.1　问题一代码

```
import jieba #导入模块jieba
filename = "留言信息.txt" #需要打开的文件名
stopwords_file = "stop_words.txt" #停用词文件名
#读取停用词，进行处理
stop_f = open(stopwords_file,"r",encoding='utf-8')
stop_words = list()
for line in stop_f.readlines():
line = line.strip()
if not len(line):
continue

stop_words.append(line)
stop_f.close

print(len(stop_words))
#读取文本留言信息，并处理留言数据
f = open(filename,"r",encoding='utf-8')
result = list()
#分词处理
for line in f.readlines():
line = line.strip()
if not len(line):
continue
outstr = ''
seg_list = jieba.cut(line,cut_all=False)
for word in seg_list:
if word not in stop_words:
if word != '\t':
outstr += word
outstr += " "
# seg_list = " ".join(seg_list)
#得出结果
result.append(outstr.strip())
```

```
f.close
#保存在另一个文件中
with open("留言after_.txt","w",encoding='utf-8') as fw:
for sentence in result:
sentence.encode('utf-8')
data=sentence.strip()
if len(data)!=0:
fw.write(data)
fw.write("\n")

print ("end")

#导入模块，模块word2vecsys
from gensim.models import word2vec
import sys提取每一行的关键词，并计算向量相似度

sentences=word2vec.Text8Corpus(u'mod.txt')
model=word2vec.Word2Vec(sentences, size=10)
print()
for i in model.most_similar(u"公司",topn=20):
print (i[0],i[1])
model.save('/text82.model')
print(model['城乡建设'])

from jieba import analyse
# 引入TF-关键词抽取接口IDF
tfidf = analyse.extract_tags
textrank = analyse.textrank

filename = "mod.txt"
# 基于TF-算法进行关键词抽取IDF
content = open(filename, 'rb').read()
keywords = tfidf(content)
print ("keywords by tfidf:")
# 输出抽取出的关键词
for keyword in keywords:
print (keyword + "/")
```

```
print ("\nkeywords by textrank:")
# 基于算法进行关键词抽取TextRank
keywords = textrank(content)
# 输出抽取出的关键词
for keyword in keywords:
print (keyword)
print("end")
print(model[keywords])

#计算词频数
import collections
#coding=utf-8
filename = "mod.txt"
with open (filename,'rb') as f:
words_box=[]
words_box2=[]
for line in f:
line.decode("utf-8")
words_box.extend(line.strip().split())
for word in words_box:
word2 = word.decode("utf-8")
words_box2.append(word2)
print("词的总数为: %s"%len(words_box2))
print("词频结果: %s"%collections.Counter(words_box2))


model.save('/text82.model')
print(model['man'])
```

1.2 问题二代码

```
## 导入模板
import jieba
import jieba.analyse
import re
import pandas as pd
```

```
import jieba.posseg as peg
import collections
import xlwt
## 文本挖掘

#第一步：数据清洗
f = open('result.txt','r',encoding='utf-8') #以读的方式打开文档
txt = f.read() #读取字符串
f.close() #关闭
txt = re.sub(', [? !?” “. 、]','',txt) #替换部分特殊字符
#print(txt)

#第二步：分词和词性
jieba.load_userdict('newdic1.txt') #在库中加入自定义分词词典jieba
seg_list = jieba.cut(txt,cut_all=False)
# seg_list = jieba.cut_for_search(txt)
txt_cut=" ".join(seg_list)
#print(txt_cut) # 精确模式输出分词结果
list = peg.cut(txt_cut)
tag_list =[]
for tag in list :
pos_word = { }
pos_word[1] = tag.word
pos_word[2] = tag.flag
tag_list.append(pos_word)
#print('词性标注：jieba',tag_list)
p = open(r'result.txt', 'r', encoding = 'utf-8') #encoding=utf-8
q = open(r'fenci_cix.txt', 'w', encoding = 'utf-8')
for line in p.readlines():
words = peg.cut(line)
for word, flag in words:
q.write(str(word) + str(flag) + " ")
q.write('\n')
f1 = open('fenci_cix.txt','r',encoding='utf-8') #以读的方式打开文档
txts = f1.read() #读取字符串
f1.close() #关闭
#print(txts)
```

```
#第三步：去停用词
stopwords = {}.fromkeys([ line.rstrip() for line in
    open('stopwords.txt','r',encoding='UTF-8') ])
final = ""
for word in txt:
if word not in stopwords:
if (word != "。" and word != ", ") :
final = final + "" + word
print(final)

with open("final.txt", "w") as f:
f.write("final") # 这句话自带文件关闭功能，不需要再写f.close()

#frequency = collections.Counter(final.split())
#print(frequency)

jieba.load_userdict('newdic1.txt')
seg_list = jieba.cut(final,cut_all=False)
# seg_list = jieba.cut_for_search(txt)
txt_cut=" ".join(seg_list)
#print(txt_cut) # 精确模式


## 导入模板
import os
import csv
import codecs
import jieba
import jieba.posseg as pseg
import re
import jieba.analyse
#### 文本挖掘
os.chdir(r'D:')
## 数据清洗
txt = open("result.txt", encoding="utf-8").read()
txt = re.sub(', [? !?” “. 、]','',txt)
```

```
## 词频统计
# 去停用词
stopwords = [line.strip() for line in open("stopwords.txt",
    encoding="utf-8").readlines()]
# 分词jieba
cixing = pseg.lcut(txt)
jieba.load_userdict('newdic1.txt')
count = jieba.lcut(txt)
word_count = {}
word_flag = {}
all = []
with codecs.open(filename='word_count_cixing.csv', mode='w',
    encoding='utf-8')as f:
write = csv.writer(f, dialect='excel')
write.writerow(["word", "count", "flag"])

## 词性统计
for w in cixing:
word_flag[w.word] = w.flag

## 词频统计
for word in count:
if word not in stopwords:
# 不统计字数为一的词
if len(word) == 1:
continue
else:
word_count[word] = word_count.get(word, 0) + 1
items = list(word_count.items())

## 按词频排序
items.sort(key=lambda x: x[1], reverse=True)

## 查询词频字典里关键字的词性
for i in range(len(items)):
word = []
```

```
word.append(items[i][0])
word.append(items[i][1])
# 若词频字典里，该关键字有分辨出词性，则记录，否则为空
if items[i][0] in word_flag.keys():
word.append(word_flag[items[i][0]])
else:
word.append("")
all.append(word)
for res in all:
write.writerow(res)

## 提取句子中的关键字及其权重python
import jieba.analyse
import random,os
import re
jieba.load_userdict('newdic1.txt') #加自定义分词字典
f_tar = open('final.txt', 'r', encoding='utf-8')
#f_tar = open('result.txt','r',encoding='utf-8')
data_tar = f_tar.readlines()

for src in data_tar:
src_temp = src.strip()
print('',src_temp)
#print(jieba.analyse.extract_tags(sentence=src_temp,topK=4,withWeight=True))
for keyword, weight in jieba.analyse.extract_tags(sentence=src_temp, topK =
    4,withWeight=True,allowPOS = ()):
print(keyword, weight)
#print(src_temp,weight)
#sentence=src_temp 为待提取的文本
#topK返回几个: TF/IDF 权重最大的关键词，默认值为。20
#withWeight是否一并返回关键词权重值，默认值为:。False
#allowPOS仅包括指定词性的词，默认值为空，即不进行筛选。:

## 导入模板
from jieba import lcut
from gensim.similarities import SparseMatrixSimilarity
from gensim.corpora import Dictionary
```

```
from gensim.models import TfidfModel
# 文本集和测试词
texts = open('result.txt', 'r', encoding='utf-8')
keyword = '市万科魅力之城商铺无排烟管道，小区内到处油烟味A'
# 、生成分词列表1
texts = [lcut(text) for text in texts]
# 、建立词典，并获得词典特征数2
dictionary = Dictionary(texts)
num_features = len(dictionary.token2id)
# 、将分词列表集转换成稀疏向量集，称作语料库3.1
corpus = [dictionary.doc2bow(text) for text in texts]
# 、把搜索词也转换为稀疏向量3.2
kw_vector = dictionary.doc2bow(lcut(keyword))
# 、4TF-模型IDF
tfidf = TfidfModel(corpus)
# 、处理被检索文本和搜索词5
tf_texts = tfidf[corpus] # 此处将语料库用作被检索文本
tf_kw = tfidf[kw_vector]
# 、相似度计算6
sparse_matrix = SparseMatrixSimilarity(tf_texts, num_features)
similarities = sparse_matrix.get_similarities(tf_kw)
for e, s in enumerate(similarities, 1):
print (e, s)
```

1.3 问题三代码

```
#确定关键词第一步分词
import jieba
with open('附件四.xlsx') as f:
document = f.read()
document_decode = document.decode('GBK')
document_cut = jieba.cut(document_decode)
result = f.join(document_cut)
result = result.encode('utf-8')
f.close()
#第二步去停用词
stopwordpath = "stop_words.txt" #引入停用词表
```

```
stopword_dic = open(stopwordpath, 'rb')
stopword_f = sotpword_dic.read() #将停用词表转换为字典
stopwordlst = stopword_f.splitlines()
stopword_dic.close()

from collections import Counter
a = [stopword_f]     #将对应的字典转换为列表
result_mi = Counter(a)
print(dict(result_mi)) #输出mi

import warnings
import pandas as pd
import numpy as np
def get_score(wi_list,data): #wi_list: 权重系数列表,
#: 评价指标数据框data
cof_var = np.mat(wi_list) #将权重转换为矩阵
context_train_data = np.mat(data) #将数据框转换为矩阵
last_hot_matrix = context_train_data * cof_var.T
last_hot_matrix = pd.DataFrame(last_hot_matrix) #权重与自变量相乘
last_hot_score = list(last_hot_matrix.apply(sum))#累加求和得到总得分

#数据标准化
from __future__ import print_function, division
#标准化方法归一化:
data0 = [(x - min(data))/(max(data) - min(data))]

def get_entropy_weight(data):
m,n=data.shape
data=data.as_matrix(columns=None) #将格式转化为矩阵dateframe
k=1/np.log(m)
yij=data.sum(axis=0)
#第二步, 计算pi
pi=data/last_hot_score
test=pi*np.log(pi)
test=np.to_num(test)
#计算每种指标的信息熵
E(T)=-k*(test.sum(axis=0))
```

```
#计算每种指标的权重
wi=(1-E(T))/np.sum(1-E(T))
wi_list=list(wi)
return(wi_list) #输出权重

#余弦相似度算法确定完整性
import numpy as np
def bit_product_sum(x, y):
return sum([item[0] * item[1] for item in zip(x, y)])

def cosine_similarity(x, y, norm=False):
#计算两个向量和的余弦相似度xy
assert len(x) == len(y), #len(x) != len(y)
zero_list = [0] * len(x)
if x == zero_list or y == zero_list:
return float(1) if x == y else float(0)
# 余弦相似算法
res = np.array([[x[i] * y[i], x[i] * x[i], y[i] * y[i]] for i in
    range(len(x))])
cos = sum(res[:, 0]) / (np.sqrt(sum(res[:, 1])) * np.sqrt(sum(res[:, 2])))

#计算复相关系数
from math import sqrt
def multipl(a,b):
sumofab=0.0
for i in range(len(a)):
temp=a[i]*b[i]
sumofab+=temp
return sumofab

def corrcoef(x,y):
R=length(x)
sum1=sum(x)
sum2=sum(y) #求和
sumofxy=multipl(x,y) #求乘积之和
#求平方和
sumofx2 = sum([pow(i,2) for i in x])
```

```
sumofy2 = sum([pow(j,2) for j in y])
num=sumofxy-(float(sum1)*float(sum2)/R)
#计算复相关系数
den=sqrt((sumofx2-float(sum1**2)/R)*(sumofy2-float(sum2**2)/R))
return num/den
return 0.5 * cos + 0.5 if norm else cos # 归一化到[0, 区间内1]
```