C 题：“智慧政务”中的文本挖掘应用

摘要

在这个信息时代，以怎样的方式去了解民意、汇集民智、凝聚民气是城市政府以及相关工作部门日益关心的重大问题之一。但随着各类社情相关的文本数据量的不断攀升，以人工的方式对群众的留言进行分类以及对热点问题的整理往往存在工作效率低下的问题。因此，本文将基于数据挖掘技术对“智慧政务“中的文本即群众的留言数、群众关心的热点问题、以及相关工作部门的解决方案数据进行内在信息的挖掘与分析。

首先，在本次数据挖掘过程中，我们首先对获取得到的留言数据利用基于 Python 的 fastText 原理对数据预处理、分词以及停用过滤操作，实现对留言数据的分类，并提升了可建模度，并使用 F-Score 对训练模型进行评价，经过不断的训练，最终得到评价值约为 0.8835。

其次，对热点问题的挖掘，我们首先对留言语料进行文本预处理，使用 TF-IDF 方法进行分词及去除停用词操作，用欧氏距离来得到相似的度量报道与话题的相关性，最后通过文本聚类 k-means 算法，把相似的高的留言加到对应的话题簇中，得到了话题簇，因此得到了排名前 5 的热点问题和相应热点问题对应的留言信息。

最后，问题 3 主要从答复意见文本的相关性、完整性和可解性、时效性和信息量等角度出发，来建立相关工作部门对留言答复意见质量的评价指标。本文运用预先相似度计算方法来计算留言主题与相关工作部门的答复意见之间的相似度， 用自动化可读性指数 ARI 来表示可解释性，ARI 的计算公式为：API=4.71*（总字符数/总字数）+0.5*（总字数/总句数）-21.43。基于主成分分析之权值计算方法，算出为接下来即将要构建的评价模型中各个评价指标的权重。其中相关性、可解释性、实效性和信息量分别所占的权重为 0.26、0.28、0.18、0.28，最后得到答复意见质量评价模型为 $Q = 0.28Words + 0.26Relevancy + 0.28Credibility + 0.18Timeliness + 0.01$。

关键词： 自然语言、fastText 原理、n-gram 特征、F-Score 评价、k-means 算法、欧氏距离、TD-IDF 方法，广义线性回归

Question C: Text Mining Application in "Smart Government Affairs"

Abstract

In this information age, how to understand public opinion, gather people's wisdom, and gather people's popularity is one of the major issues that the city government and related work departments are increasingly concerned about. However, as the amount of text data related to various social conditions continues to rise, there is often a problem of low work efficiency in manually categorizing the masses' messages and sorting out hot issues. Therefore, based on the data mining technology, this article will mine and analyze the intrinsic information of the text in the "smart government affairs", that is, the number of messages of the masses, the hot issues that the masses care about, and the solution data of the relevant work departments.

First of all, in this data mining process, we first use the Python-based fastText principle to preprocess the data, segment the words, and disable the filtering operation on the obtained message data, classify the message data, and improve the modelability And use F-Score to evaluate the training model. After continuous training, the final evaluation value is about 0.8835.

Secondly, for the mining of hot issues, we first perform text preprocessing on the message corpus, use the TF-IDF method to perform word segmentation and remove stop words, and use Euclidean distance to obtain similar measurement reports and topic relevance, and finally pass The text clustering k-means algorithm adds similar high messages to the corresponding topic clusters to obtain topic clusters. Therefore, the top 5 hotspot questions and the message information corresponding to the corresponding hotspot questions are obtained.

Finally, from the perspective of the relevance, completeness and solvability, timeliness and amount of information of the reply opinion text, to establish related work The evaluation index of the quality of the department's response to the message. This article uses the pre-similarity calculation method to calculate the similarity between the subject of the message

and the reply of the relevant work department, and uses the automated readability index ARI to indicate the interpretability. The calculation formula of ARI is: API = 4.71 * Number / Total Words) + 0.5 * (Total Words / Total Sentences) -21.43. Based on the weight calculation method of principal component analysis, the weight of each evaluation index in the evaluation model to be constructed next is calculated. Among them, the weights of relevance, interpretability, effectiveness and amount of information are 0.26, 0.28, 0.18 and 0.28 respectively, and the quality evaluation model of the final opinion is Q = 0.28Words + 0.26Relevancy + 0.28Credibility + 0.18Timeliness + 0.01 .

Keywords: natural language, fastText principle, n-gram features, F-Score evaluation, k-means algorithm, Euclidean distance, TD-IDF method, generalized linear regression

evaluation, k-means algorithm, Euclidean distance, TD-IDF method, generalized linear regression

目录

1 引言

1.1 研究背景

自然语言构成的文本中往往包含了丰富的信息，但是这些自然语言描述的信息是提供给人阅读理解，计算机无法组织里面的有效信息加以利用。一般的解决办法是人工直接从文本中提取信息，或者利用计算机程序通过自然语言特征抽取特定信息。如何让计算机更好的自动抽取文本信息成为急需解决的问题。中文文本信息抽取成为自然语言处理及文本挖掘领域的一个研究热点。

文本信息抽取主要分为实体抽取、实体关系抽取等部分，目前大多采用机器学习，尤其是基于概率统计的机器学习方法来解决这些问题。主要分为有指导(Supervised)和弱指导(Weakly Supervise)的学习方法。大多数自然语言处理问题面对的是一般领域语料，大多采用有指导的学习方法，需要费时费力的标注训练集，训练集的优劣直接决定了最终学习模型的好坏。然而信息抽取任务针对的往往是特殊领域语料，基于一般领域语料所总结出的抽取内容往往不能很好解决特殊领域问题。所以需要利用机器学习方法快速构建特殊领域文本信息抽取系统。

本文针对群众留言分类、热点留言以及相关部门回复方案评价问题实现了该方法，与直接通过模板提取信息相比，本文提出的方法大大提高了准确率召回率，以及减少了大量人工干预，建立模板的工作工作量。并且具有很好的扩展性，可以做到迅速构建系统应对新的中文文本抽取任务。

2 挖掘目标

本次的建模的目标是利用来自互联网公开来源的群众问政留言记录，及相关部门对部分群众留言的答复意见，采用 fastText 原理对文本信息进行分类，并通过不断的训练模型，调节相关参数，最终使得将不同的留言换分到不同的工作管理类别中，提高相关工作部门工作的效率。

对文本进行基本的机械预处理、中文分词、停用词过滤后、建立话题簇，对热点问题进行归类，得出当前的热点信息，以便相关部门针对性地解决实时问题，提高人民幸福指数。

实现对文本数据的倾向性判断以及所隐藏的信息的挖掘并分析，以期望得到有价值的内在内容。

3 模型构建过程及结果分析

3.1 问题 1 分析方法与过程

3.1.1 流程图

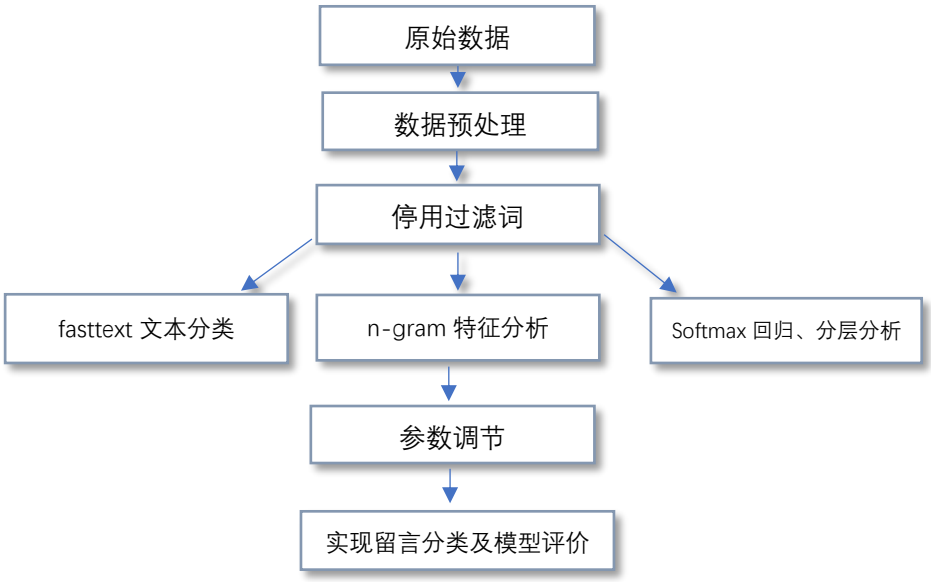

图 1 问题一流程图

3.1.2 数据预处理

3.1.2.2 机械压缩取词

由于群众留言信息数据中，有些数据质量可能存在参差不齐、没有意义的情况

3.1.3 留言文本分词

在中文中，只有字、句和段落能够通过明显的分界符进行简单的化界，而对于“词”和“词组”来说，它们的边界模糊，没有一个形式上的分节符。因此，进行

文本挖掘时，首先对文本分词，即将连续的字符按照一定的规范重新组合成词序列的过程。

问题 1 使用 Jieba 方法对留言文本分词，基于 Jieba 分词包，其运用了数据结构里的 trie（前缀数或字典树），能够对词语进行高效的分类。Trie 的原理如图 2 所示：

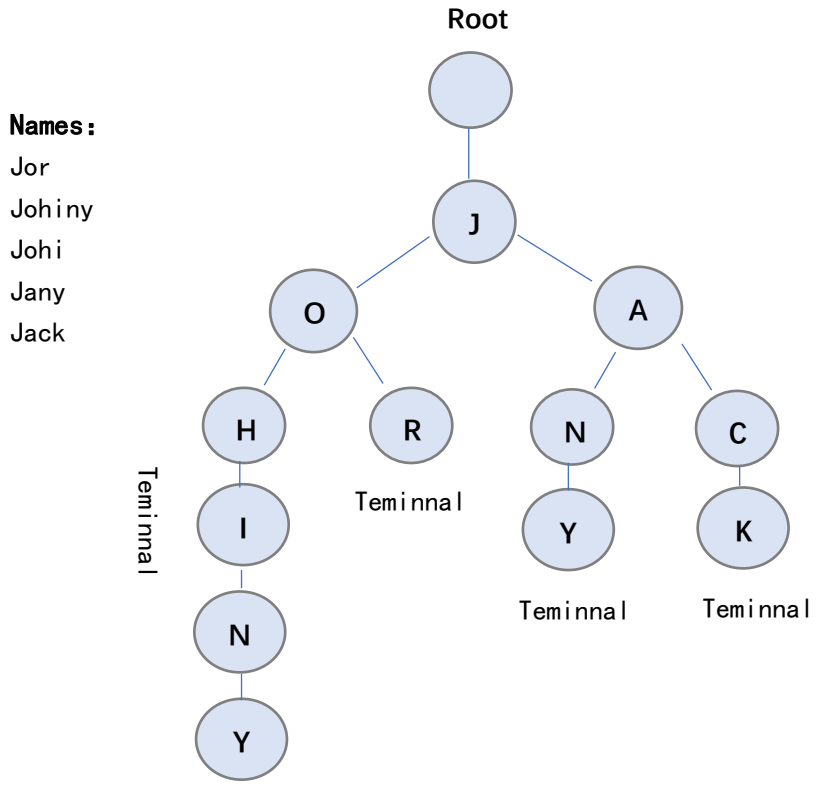

图 2 Trie 原理演示图

如上图 2 所示，比如我们有 Johiny、Jor、Jany 和 Jack 四个名字，假设我们要让计算机查找名字 Jack 是否存在，则 trie 会从上至下的搜索、每一次判定一个字母、如果某个特定的节点（node）的下一个节点（child node）不在符合搜索压迫求，那么搜索就会停止，从而使得效率大大的提高。

与此同时，在文本信息中，仅仅以 trie 原理进行分词会避免不了双重理解词语结合的情况，于是 trie 与有向无环图（DAG）的结合运用，可以高效的解决这个问题，其运用原理举例如图 3 所示：

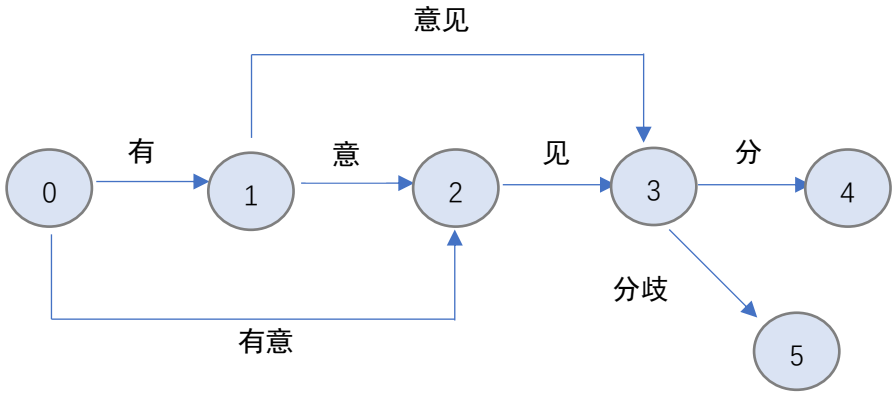

图 3　trie 与有向无环图（DAG）结合原理图

通过设定，计算机自动识别出了两种分词方法，分别是‘有/意见/分歧/’和‘有意/见/分歧’。

由于使用 tire 与有向无环图结合原理对文本进行分词，结果为一个句子有多种分词方式。对比于 Jieba 分词细分的三种模式，分别为：精确模式、全模式和搜索引擎模式，我们经过不断的测试和训练和结合我们留言文本内容，最后选择使得这个句子出现概率最大的切分组合。对于留言文本内容分词后，Jieba 的精确模式对留言文本进行分词的效果最佳。 最终我们对留言文本分词的部分结果如图 4 所示：

珠泉 开发区 港嘉 娱乐城 水压 一年 自来水 城管 时候 才能 协商 港嘉 娱乐城 住户 J10　县华邦 小区 2014 西南 物业 进住 以来 凭着 西南 物业 老板 谢立 J10　势力 加 金盾 金街 开工 解释 天气 原因 设立 公共 账号 原因 据长 居安 任县 城区 网民 透露 安宜人 广场 安置 地是 老鼠 切村 为了 宜人 广场 百通 汽车站 整体 搬迁 安置 小区 至今 区长　　棚 改户 业主 代表 五里 牌 棚户区 工程 进展 汇报 2012 年初 开工 至今 二年 严 书记 房价 动辄 上百万 公积金 买房 需要 支付 百分之五十 甚至 六十 首付 对于 家庭 K2 珍珠 北路 139 潇楚佳苑 小区 居民 艾 恩米 国际 幼儿园 小区 栋 前 小区 公共绿地 2018 投入 桃川 棚 改 资金：第一期 棚改 资金 一亿一千多万 第二期 棚改 资金 千 多万 K2 珍珠 北路 139 潇楚佳苑 小区 居民 艾 恩米 国际 幼儿园 小区 栋 前 小区 公共绿地 市金 水岸 陆续 有人 装修 居住 开发商 开通 燃气 何时 开通
K2 区创 发城 存在 违建 现象 烂尾楼 存在 几年 拆除 导致 消防通道 消防车 居民 存在 老 建筑 下岗职工 大半辈子 住 七层 坡 社区 旁边 2010 左右 老板 征收 50 ㎡ 房屋 通过　　K1 光明 新路 人行道 绿化 本是 好事 中午 光明 新路 香桂茗苑 一棵 已栽种 多年 各级：　　K2 珍珠 南路 二 标段 断头 路 居民 2008 我家 房屋 K2 人民政府 纳入 征收

图 4 留言文本分词结果

3.1.4 停用过滤词

经过中文分词这一步骤，将初始的文本处理成为词的集合，即$d = (\mu_1; u_2; , \ldots, u_n)$，其中 n 为文本 d 中出现词语的个数。但是文本中含有对文本含义表达无意义的词语，这些词的存在及其普遍，且记录这些词在每一个文档中的数量需

要很大的磁盘空间，比如文本中的一些副词、语气词以及一些无实际含义的实词，应进行删除，以消除它们对文本挖掘工作的不良影响。

对于虚词，比如文本中的“了”、“啊”、“无论”、“比如”等，特殊符号如“#”、“γ”、“μ”、“ϕ”等，英文中的“is”、“are”、“the”、“that”等。于此同时，由于在不同文本的应用中，构建的停用词表对文本数据分类的精确度以及维度有着不同程度的影响。

因此我们结和中文分词所分出的词的集合进行人工选取拟定停用此表，我们选取分类词频中前 200 的词，再通过统计这些分类在其他分类中出现的情况，即一个分词词频在各个类标签中出现词频是 200，且该分词在超过四个类中同时出现，我们则将此定义为停用词，具体停用此表详见附录一

使用停用词表的效果示例如下：

原始留言：关于预防先天缺陷的建议

结果：预防 先天 缺陷 建议

3.1.5 方法实现过程

3.1.5.1 n-gram 特征

在文本特征提取中，常常能看到 n-gram 的身影。它是一种基于语言模型的算法，基本思想是将文本内容按照字节顺序进行大小为 N 的滑动窗口操作，最终形成长度为 N 的字节片段序列。在本文问题 1 中，我们经过对数据的不断测试之后，得到的真是效果和时间空间的开销权衡之后，得出 2-gram 模型最适用且最为合理，即假设我们有有 m 个词组成的序列（或者说一个句子），根据链式规则，可得到整句的概率$P(\omega_1,\omega_2,\dots,\omega_3)$，即

$$P(\omega_1,\omega_2,\dots,\omega_3)=P(\omega_1)*P(\omega_2|\omega_1)*P(\omega_3|\omega_1,\omega_2)\dots.P(\omega_m|\omega_1,\omega_2,\dots\omega_{m-1}) \quad (1)$$

此时由于 n=2，因此所构建的二元模型（bigram model）为：

$$P(\omega_1,\omega_2,\dots,\omega_m)=\prod_{i=1}^{m}P(\omega_i|w_{i-1}) \quad (2)$$

比如任意选取群众问政留言数据中某一条分析如下：

A 市何时能实现冬季集中供暖

相应的 bigram 的特征为：A 市 市何 何时 时能 能实 实现 现冬 冬季 季集 集中 中供 供暖

相应的 trigram 特征为：A 市何 市何时 何时能 时能实 能实现 实现冬 现冬季 冬季集 集中供 中供暖

经过以上分词和特征提取步骤，最终得到留言文本部分词频数据如图 5 所示：

教育文体		劳动和社会保障		环境保护		城乡建设	
,	34207	,	33566	,	16260	,	32699
的	21219	的	19780	的	8743	的	17416
。	10509	。	10749	。	4575	。	9175
	8674		10145		3946		7544
	6261	你	8515	、	2640		5118
、	6083	我	5524		2538	、	5028
了	4787	了	5132	是	1828	了	4360
是	4707	、	5059	在	1818	是	3818
在	3980	我们	4967	了	1711	在	3677
学校	3568	是	4609	我们	1671	我们	2998
我	3293	年	4540	!	1235	!	2371
?	2913	在	4284	和	1230	?	2369
我们	2662	月	2921	*	1175	我	2364
!	2534	?	2744	污染	949	市	2331
和	2519	和	2602	都	918	和	2275
不	2435	*	2467	严重	881	不	2068
学生	2352	!	2389	村民	872	业主	1909
市	2293	不	2353	我	871	有	1822
有	2213	市	2345	市	854	都	1818
都	2017	:	2176	居民	851	年	1768
教师	1928	没有	2068	有	839	小区	1729
也	1895	工作	2019	?	793	也	1704
"	1856	也	1988	:	792	:	1698
“	1854	都	1887	不	768	没有	1649
:	1755	有	1781	也	728	就	1519
年	1753	为	1691	县	715	*	1391
教育	1696	公司	1679	对	714	月	1386
就	1677	)	1679	没有	683	县	1292

图 5 分词频数统计结果

3.1.5.2 Softmax 回归

Softmax 回归（Softmax Regression）又被称作多项逻辑回归（multinomial logistic regression），它是逻辑回归在处理多类别任务上的推广。

在逻辑回归中，我们有 m 个被标注的文本：$\left\{\left(\left(x^{(1)},y^{(1)}\right),\ \ldots,\ \left(x^{(m)},y^{(m)}\right)\right)\right\}$，其中，$x^{(i)} \in R^n$。由于类标是二元的，所以我们有$y^{(i)} \in \{0,1\}$，我们假设（hypothesis）有如下形式：$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$代价函数(cost function)如下：

其中 1{•}是指示函数，即 1{true}=1，1{false}=0

$$J(\theta) = -\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right) \log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] \tag{3}$$

在 Softmax 回归中，类标是大于 2 的，因此我们的训练集$\{((x^{(1)},y^{(1)}),\ldots,(x^{(m)},y^{(m)}))\}$中，。$y^{(i)} \in \{1,2,\ldots,K\}$。给定一个测试输入 x，我们输入一个 K 维的向量，向量内每个元素的值表示 x 属于当前类别的概率。

在标准的 Softmax 回归中，由于要计算$y = j$时 Softmax 的概率：$P(y = j)$,因此需要对所有的 K 个概率做归一化。于是使用 Softmax 分层示例如图 6 所示：

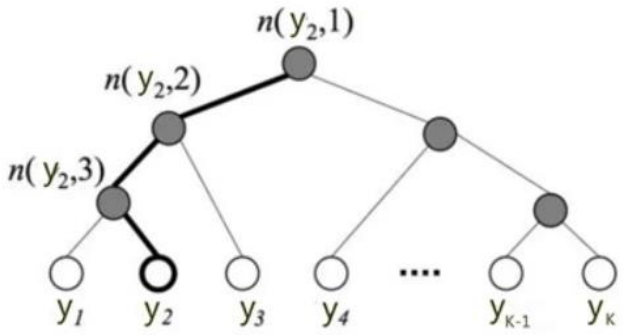

图 6 Softmax 分层示例

3.1.5.3 fastText 模型构架

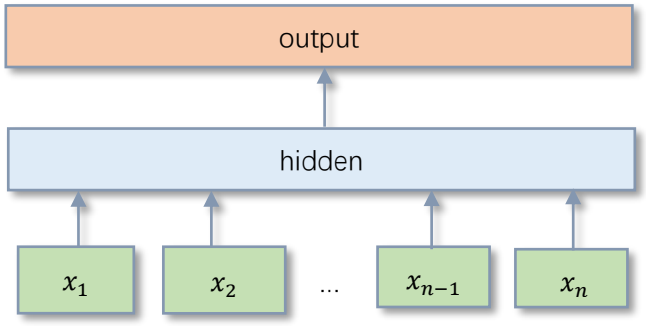

图 7 fastText 模型构架

fastText 模型有三层：输入层、隐含层、输出层。

输入层： 输入层输入的是一批文档，每个文档由一个词汇索引序列构成。例如在处理问题 1 时，[10 30 80 1000] 可表示“A 市 渔业路 洒水车 扰民”这个短文本，其中“A 市”、“渔民路”、“洒水车”、“扰民”在词汇表中的索引分别是 10、30、80、1000；

隐含层：隐含层对一个文档中的所有文本信息的向量进行叠加平均；
输出层：输出的是一个特定的 target；
在输出时，fastText 采用了 Softmax 很大程度上降低了模型训练的时间。
模型搭建遵循以下步骤：

fastText 的代码结构以及各模块的功能如图 8 所示：（代码详见附录二）

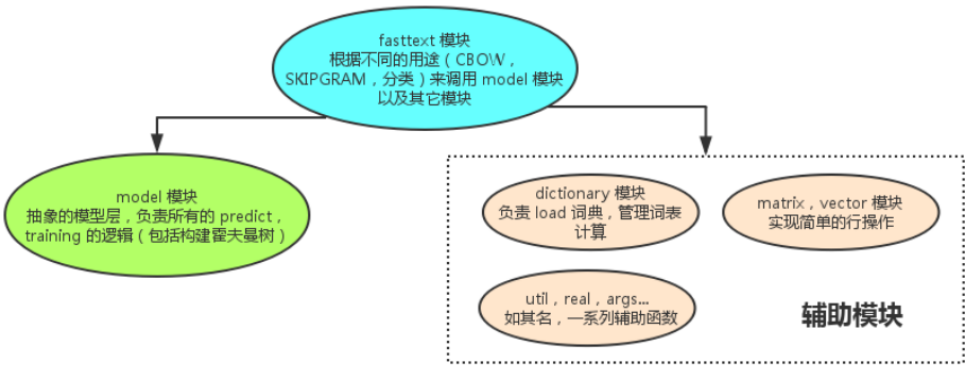

图 8 fastText 代码结构图

训练数据格式一行为一个句子，每个词用空格分隔，如果一个词带有前缀“__label__”，那，那么它就作为一个类标签，在文本分类时使用。于此同时，经过对数据选取以及对模型参数的不断调节并加以训练，最终得到文本分类的最佳模型，并使用 F-Score 对分类模型进行评价，最终评价值约为：0.8835

其中，F-Score 评价公式为：

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i + R_i} \tag{4}$$

其中，P_i为第i类的查准率，R_i为第i类的查全率。
使用 fastText 分类结果以及 F1 值结果如表 1 所示：

表 1　留言分类结果

	P	R	F
劳动和社会障	0.937500	0.937500	0.937500
交通运输	0.900000	0.725806	0.803571
城乡建设	0.829268	0.880829	0.854271
卫生计生	0.964286	0.830769	0.892562
教育文体	0.891026	0.932886	0.911475
商贸旅游	0.812500	0.873950	0.842105
环境保护	0.972973	0.915254	0.943231

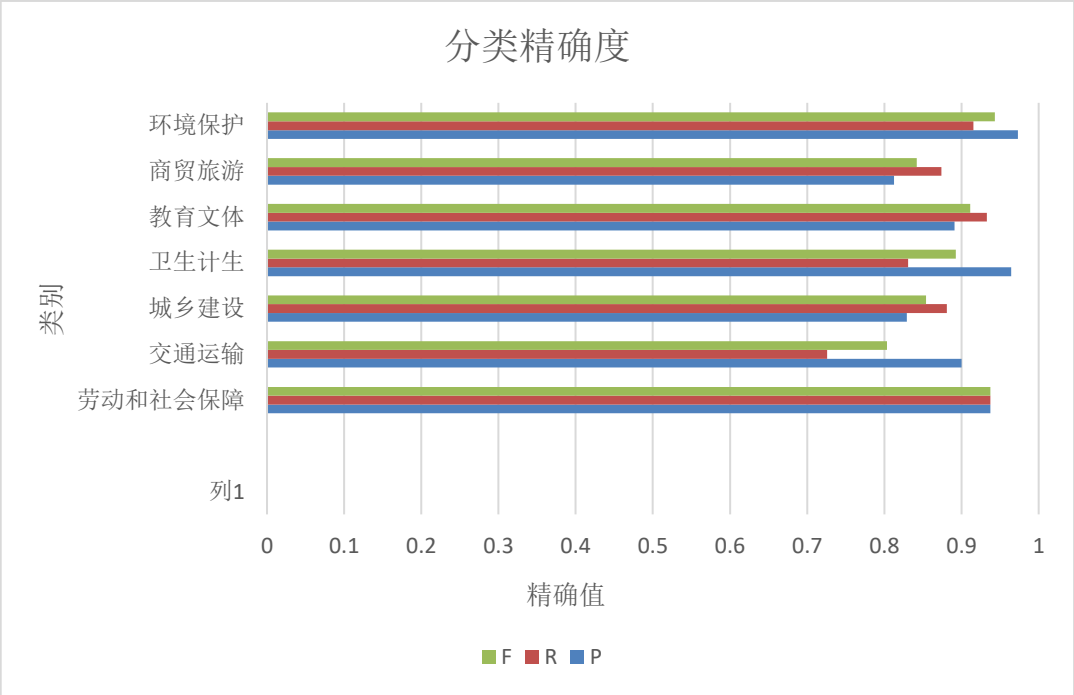

图 9　留言分类结果 FGP 值

3.2 问题 2 方法与过程

3.2.1 问题分析：

对于话题热度影响指标，在传统的话题中大部分是考虑了新闻的标题、正文等

文本信息特征。然而，对于网络留言是有用户的参与，如评论、点赞等。针对新闻报道数据来说，首先，一个话题参与的人数越多，即该话题是人们关注并且讨论较多的，说明该话题在当前时间的是受人们关注的。其次，每个热门话题都经历一个生命周期，即每个话题的“热度”是在给定一段时间内发展的。因此，话题参与人数、点赞人数、反对人数、留言持续时间等可作为话题热度的一个影响指标，体现了用户的参与程度。

总之，再留言数据的背景下，一个“热点”话题具备以下的特点：

（1） 它有较强的持续性，即被定义为一段时间内多次被人们提及到的事件。

（2） 它的受欢迎程度（即热度）是随时间变化的。

（3） 它受人们关注度较高，对于此人们会有较多的赞成或是反对的声音。

3.2.2 话题发现基本流程图

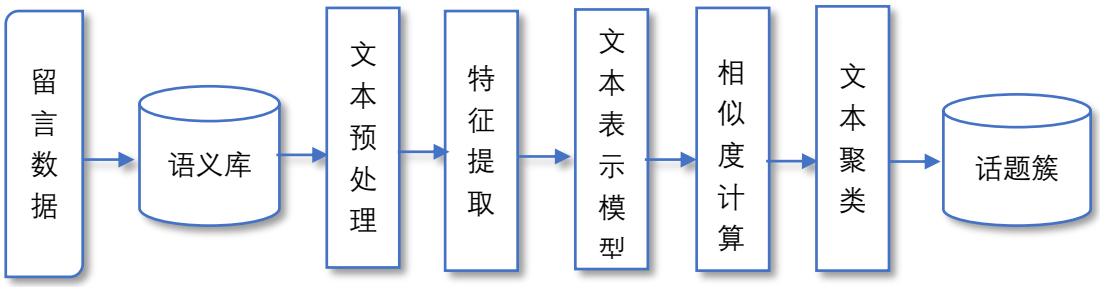

图 10　留言话题发现处理流程图

从图 10 中可知，留言话题发现从文本预处理到文本聚类共有五个主要步骤。首先留言数据公开已知，然后将所得的数据保存到奥文本语料库中；其次，对留言语料进行文本预处理，使用 TF-IDF 方法进行分词及去除停用词操作，并从预处理后的文本数据中提取特征此项建立文本表示模型；然后，用相似的度量报道与话题的相关性；最后通过文本聚类算法，把相似的高的留言加到对应的话题簇中，这样就得到了话题簇。

3.2.3 文本预处理

3.2.3.1 留言文本分词

同样的，我们使用问题 1 中的 Jieba 之精度确认模式对留言文本数据进行分词，然后构造通用词去除没有实际意义的词语。具体操作流程已经在问题 1 中进行详细描述，在此处不在重述。

最终得到留言分词结果如图 11 所示：

中 什么 5 这个 去 已 尊敬 路 这样 时 请问 有关
以 县 您好 还是 6 一 过 2018 没 就是 于是否
无法 10反映 在 他们 本人 很 个 号 把 通过 导致
好 不是 需要 8 如果 想 造成 目前 开始 却 因为 为什么
任何 且 回复 此 谢谢 老百姓 其他 知道 建议 很多 恳请 时候
这些 请求 正常 内 着 更 得到 关于 国家 行为 每天 非常 重视
事情 那 期间 国际 所 提供 一名 完全 必须 具体 发展 群众
打 全部 里 社会 城 如此 各种 所以 基本 根本 两个
诉求 市 答复 呢 位于 其中 三 跟 他 难道 增加 考虑 合理
可能 近 一年 出 同意 不要 如 您 左右几个 米 一条 继续 理由
承诺 一次 休息 标准 信息 主要 沟通 关注 事实真相 第十六条
考虑一下 知悉 国家有关 第一条 需在 一万多 没什么 听听 2019.1
总体 身边 故事 自相矛盾 仅为 两三点 更有甚者 写信 日益 谁给了
一意孤行 睁一只眼 闭一只眼 以往 多天 长远 当晚 咀 三月份 初期

图 11 留言分词结果

使用停用词，去掉留言文本中的虚词即副词、语气词等与无实际意义的词后，得到部分结果如图 12 所示：

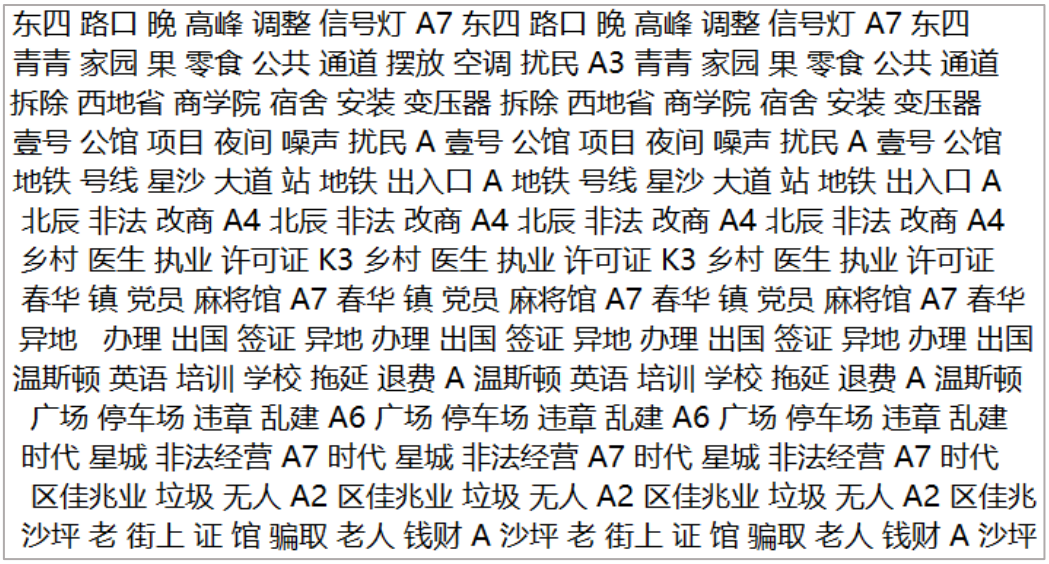

图 12 留言文本去停用词结果

3.2.4 留言信息特征提取

本文采用向量空间模型表示留言文本信息。将一个留言文本数据d表示n维向量，即$d=(t_1,w_1;t_2,w_2;\dots;t_i,w_i;\dots;t_n,w_n;)$，其中：$n$是$d$中的特征总数，$t_i(1<i<n)$为$d$中对应的权重值，表示它在$d$中重要的程度。

本文用 TD-IDF 计算特征词项权重，具体操作如下：

该操作的主要目的有两个，第一，为了提高程序的效率，提高运行速度；第二，所有词汇对文本分类的意义是不同，一些通用的、各个类别都普遍存在的词汇对分类的贡献小，为了提高精度，对于每一类应去除那些表现力不强的词汇，筛选出针对该类的特征项集合，采用了 TFIDF 方法来进行特征选择。该方法所用的主要公式如下所示，

TD-IDF 的计算：

以及基于 TF-IDF 算法对留言文本进行分词以及关键字的提取，其中经过分词后统计词频所用的计算原理如下：

假设文档集合

$$T=\{n|t_n,n>1\} \tag{5}$$

且

$$TF-IDF=TF*IDF \tag{6}$$

其中，TF（term frequency，TF）：词频，某个给定词语在该文件中出现的次数，计算公式为：

$$TF_w = \frac{\text{在某一类中词条 w 出现的次数}}{\text{该类中的词条数目}} \quad (7)$$

IDF（inverse document frequency，IDF）：逆文件频率，如果半酣词条的文件越少，则说明词条具有很好的类别区分能力，计算公式为：

$$IDF = \log\left(\frac{\text{语料库中总文档数}}{\text{包含词条 w 的文档数} + 1}\right) \quad (8)$$

根据上面的计算我们可以算出文件，单词条w的 TD－IDF 权值 W[i][j]= TD(j)*IDF(j)。其中 i 为文件集合 T 中的一个文件，而 j 是文件集合 T 中的一个单词。

通过对文件集合 T 的计算我们可以得到二维数组(矩阵)W[i][j].最终得到的特征此项如附录二，部分词项如图 13 所示：

第0类关键词： 渣土 车 施工 夜间 噪音 扰民 A3 作业 工地 城管 凌晨
第1类关键词： 物业 投票 委员 委员会 物业公司 管理 社区 招标 委会 业委会 大会
第2类关键词： 幼儿 幼儿园 A3 教育局 保利 A4 就读 街道 公立 孩子 北辰
第3类关键词： 工资 员工 民工 农民 农民工 拖欠 基层 劳动 社保 项目 包工
第4类关键词： 车库 物业 地下 车辆 停车 服务 管理 质量 地面 车位 收费
第5类关键词： 停车 车道 交警 乱 道路 浪琴 湾 车场 停车场 出行 溪湖
第6类关键词： 搅拌 搅拌站 新城 丽发 噪音 A2 区丽发 环境 生活 污染 灰尘
第7类关键词： 摊贩 经营 摆摊 城管 市场 环境 凌晨 生活 店外 门口 流动
第8类关键词： 物业 曙 停电 光 学校 职校 供电 生活 管理 停水 检修
第9类关键词： A1 立交 立交桥 区路 路口 新建 左转 赤岭 车道 右转 交通
第10类关键词： 商业 东 六路 地块 泉塘 中心 用地 综合 合体 综合体 工业
第11类关键词： 景园 滨河 苑 市伊 购房 购房者 车位 铁路 违规 捆绑 集团
第12类关键词： 广告 贴 免费 活动 参与 地铁 宣传 预存 报名 二号线 身份
第13类关键词： 社区 尖山 社员 拆迁 居委 委会 居委会 施工 干部 集体 房子
第14类关键词： 保安 地铁 西湖 文化 文化园 外省 巡逻 偷 管理 维权 安保
第15类关键词： 地铁 潭 长株 城铁 上班 出行 交通 下班 路口 公交 西路
第16类关键词： 社保 系统 保险 生育 窗口 社保局 养老 购房 记录 单位 养老保险
第17类关键词： 高速 收费 收费站 A8 绕城 免费 停车 通道 出口 金桥 车道
第18类关键词： 交房 物业 施工 标准 消防 验收 质量 宣传 合同 建筑
第19类关键词： 噪声 施工 扰民 凌晨 噪音 环境 A3 工地 污染 声音 作业

图 13 提取词项关键词

3.2.5 话题的表示模型

3.2.5.1 k-means 聚类算法

对于留言文本语料进行话题发现时，我们无法预测会有多少个留言话题，以及

何时又出现新的话题。因此，这个领域研究也等同于对无监督、无指导的聚类算法分析。聚类算法就是无监督的机械学习方法，将数据集划分为不同的类簇。将每个族看成是一个话题， 然后运用 k-means 聚类方法采用距离作为相似性的评价指标，即认为两个对象的距离越近，其相似度就越大。该算法认为簇是由距离靠近的对象组成的，因此把得到紧凑且独立的簇作为最终目标。其中，k 个聚类具有以下特点：各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开。

3.2.5.2 k-means 聚类的迭代过程:

1.随机选取 k 个文件生成 k 个聚类 cluster,k 个文件分别对应这 k 个聚类的聚类中心 Mean(cluster) = k ;对应的操作为从 W[i][j]中 0～i 的范围内选 k 行（每一行代表一个样本），分别生成 k 个聚类，并使得聚类的中心 mean 为该行。

2.对 W[i][j]的每一行，分别计算它们与 k 个聚类中心的距离（通过欧氏距离）distance(i,k)。

3.对 W[i][j]的每一行，分别计算它们最近的一个聚类中心的 n(i) = ki。

4.判断 W[i][j]的每一行所代表的样本是否属于聚类，若所有样本最近的 n(i)聚类就是它们的目前所属的聚类则结束迭代，否则进行下一步。

5.根据 n(i) ，将样本 i 加入到聚类 k 中，重新计算计算每个聚类中心（去聚类中各个样本的平均值），调到第 2 步。

3.2.5.3 中心点的选择

k-means 算法的能够保证收敛，但不能保证收敛于全局最优点，当初始中心点选取不好时，只能达到局部最优点，整个聚类的效果也会比较差。我们采用以下方法来确定 k-means 中心点：

1、选择彼此距离尽可能远的那些点作为中心点，对于 sklearn 中：
Km=KMeans（init=’k-means++’）

2、先采用层次进行初步聚类输出 k 个簇，以簇的中心点的作为 k-means 的中心点的输入。Km=KMeans（init=’random’）

3、多次随机选择中心点训练 k-means，选择效果最好的聚类效果

3.2.3.3 K 值的选择的依据

$$SSE = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2 \qquad (9)$$

其中，C_i是第 i 个簇，p 是C_i中个的样本点，m_i是C_i的质心（C_i中所有样本的均值），SSE 是所有样本的聚类误差，代表了聚类效果的好坏。

3.2.5.4 每个点到中心点的欧氏距离

欧几里得距离的定义如下：

$$d(X,Y)=\sqrt{（x_1-y_1）^2+（x_2-y_2）^2+\cdots+（x_n-y_n）^2} \quad （10）$$

其中x_i，y_i是文件单词的 TD-IDF 值和聚类中心的 TD-IDF 值，就可以算出与 k 个聚类中心的距离。对 W[i][j]的每一行，分别计算它们最近的一个聚类中心的 n(i) = k_i，并且判断 W[i][j]的每一行所代表的样本是否属于聚类。

基于簇内误差平方和如公式（9），使用肘方法确定簇的最佳数量，该方法的基本理念就是找出聚类偏差骤增的 k 值。最后，经过不断的测试和训练，最终我们得到当 k=1500 时，留言文本的聚类效果最佳。 最终的聚类结果以及代码实现如附录三。

3.2.6 热值计算

基于以上步骤，我们得到留言文本的聚类结果，即是把留言文本数据中相似的留言话题聚类，然后将其划分为同一类。与此同时，对于聚类后的不同话题，需对其进行话题热点的确定。在现实生活中，热点问题的确定伴随着诸多因素的作用，但在的已知数据中，我们将话题的点赞数、反对数、话题留言人数量以及该话题的活跃天数（即持续时间）作为热点话题的印象因子。再确定各因子间对一个话题的影响力度，从而确定各个因子的权值。在这里，我们进入符号：

Hot:话题

α:点赞数

β:反对数

γ:话题留言数量

δ:活跃天数

由于留言是人们对生活各种问题的反映，希望将问题反映到相关工作部门，提高工作效率。因此，在一个话题中，一方面是点赞数即代表着人们支持的态度，点赞数α越多，意味着人们更迫切相关工作部门尽快给予回复与解决方案，即点赞人数α对话题热点值是起到正向促进作用；另一方面，反对人数β越多，说明该话题的实际意义不大，则反对人数β对话题热点值起到反向削弱作用；与此同时，话题留言数量越多γ则说明人们关注点集中，对话题热点值是起到正向促进作用，且活跃天数δ持续越长，说明话题人们持续关注，对话题热点值是起到正向促进作用。

于是有，话题热点值计算公式如下：

$$Hot = 0.3\alpha - 0.1\beta + 1.2\gamma + 0.8\delta \quad (11)$$

最终得到话题的聚类结果与话题热值的具体代码实现过程如附录三。

3.2.7 文本聚类话题提取

根据得到的聚合的话题类别，我们参考类似于数据附录 1 中二级标题与三级标题，再结合留言数据的文本内容，经过人工的不断提炼，得到全部话题分类之主题提取结果及具体实现代码如附录二所示，其中，排名前五的热点话题相关内容如表 2 与图 14 所示：

表 2 排名前五的热点

热度排名	问题 ID	热度指数	时间范围	地点人群	问题描述
1	163	730	2017.06.08 至 2019.11.27	A 市经济学院学生	学校强制学生外出工作和实行问题
2	197	714.4	2019.04.11 至 2019.11.22	A 市 A3 区	购房配套入学与孩子受教育权利问题
3	444	630.3	2019.08.19 至 2019.08.19	A 市 A5 区	小区住房安全保障与租房制度混乱一系列问题
4	547	395.8	2019.01.12 至 2019.-09.05	A 市 A4 区	绿地海外滩小区高铁对周边影响问题
5	1332	394.4	2019.01.11 至 2019.07.08	A 市	非法经营车贷并创造车贷诈骗案问题

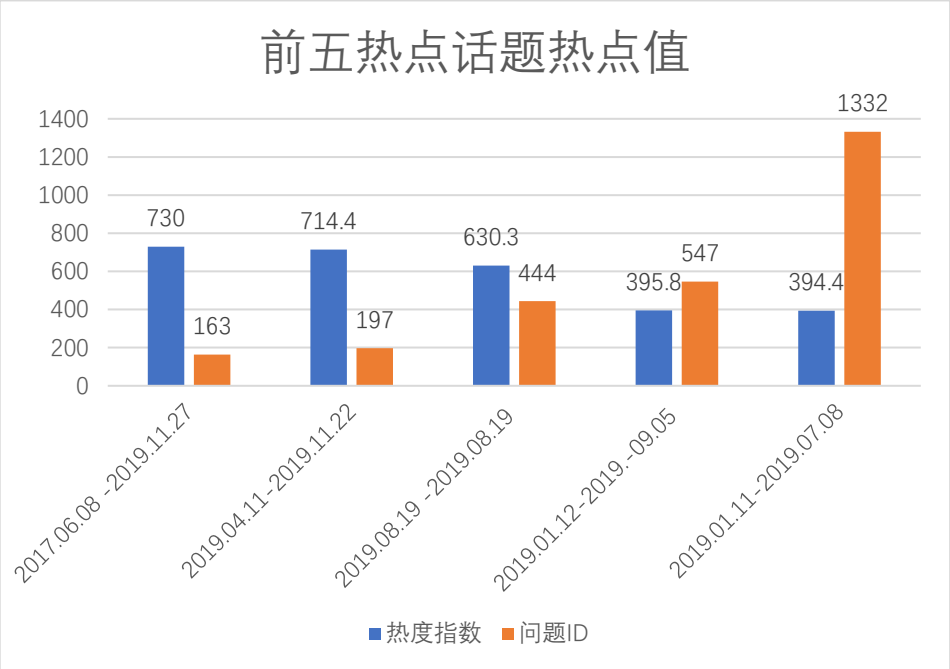

图 14 排名前五的热点

根据表 2 与图 14 可知，排名第一的热点问题是 A 市经济学院被学校强制学生外出工作和实行问题，其问题 ID 为 163，话题热度值为 730；排名第二的热点问题是购房配套入学与孩子受教育权利问题，其问题 ID 为 197，话题热度值为 714.4；排名第三的热点问题是小区住房安全保障与租房制度混乱一系列问题，其问题 ID 为 444，话题热度值为 630.3；排名第四的热点问题是绿地海外滩小区高铁对周边影响问题，其问题 ID 为 547 其话题热度值为 395.8；排名第五的热点问题是非法经营车贷并创造车贷诈骗案问题，其问题 ID 为 1332，话题热度值为 394.4；

3.3 问题 3 方法与过程

3.3.1 问题分析

在日常生活中，人们渴望对“政务”的留言得到政府以及相关工作部门的合理回复和解决方案与政府以及相关工作部门希望能够给人民一个好的答复意见是一个相互的过程。那么，在这个信息发达的时代，人们利用留言的方式向政府以及相关工作部门吐露“心声”，但是对于政府以及相关工作部门的回复，我们需要有一个评价标准以判断政府以及相关工作部门所给答复意见的质量。一方面这不但确保人民提出的问题是否可以得到有效的解决，另一方面是又可以作为一个评价政府以及相关工作部门的工作效率。

对于如何构建答复意见质量评价指标与模型，本文主要从以下几个步骤进行分析与处理，首先针对相关部门对留言所给的答复意见选取答复意见数据特征，然后根据答复意见文本提取其他主要特诊，并通过各种计算和编程得到适合模型的指标变量以及答复意见数据质量评价指标，最后建立相关工作部门对留言答复意见质量的评价指标。整个操作过程路程图如图 15 所示：

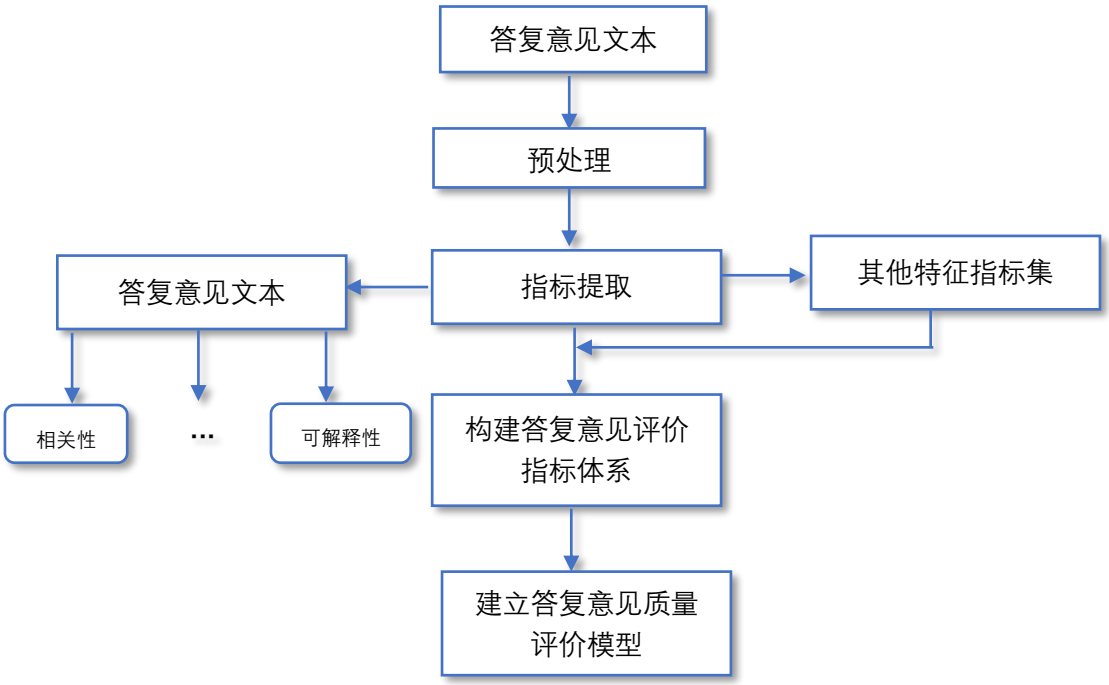

图 15 构建答复意见质量评价指标与模型流程图

3.3.2 预处理

由于答复意见是以文本的形式存在的，其复杂度相比于数值型要复杂的多。因此我们根据整理好的答复意见内容，对数据经过一系列的计算和编程处理，提取出于留言主题相关度和答复意见一致性等指标因素。于此同时，为保证数据的质量，降低一些无关数据对我们实验结果精确率的影响，因此我们需要对以获得的数据进行预处理和一些特殊处理，以达到优化数据的效果。其中预处理的过程主要包括以下几个方面：

（1） 对于回复答复意见重复的数，保留重复答复意见中的一条；

（2） 删除无实际意义的答复意见数据

（3） 使用停用词剔除答复意见文本内容中的无用词，比如语气词、介词、副词等无实际意义的词；

3.3.3 指标提取

指标提取是对答复意见数据进行分析的一项重要步骤。根据留言的内容，即是从人们留言所反馈的问题意见中提取特征。对于本文在指标提取中发现的一些无关指标（引用）不在此处列出，本文对于答复意见文本信息中提取的主要特征指标有相关性、完整性、可解读等。对于以上特征指标我们通过以下不同的方式获得：

3.3.3.1 相关性

对于答复意见集中每一条答复对应语料库中的一个问大哥，通常用向量的形式来表达，由于两个相似的文档会有相似的主题，因此可以通过计算文档之间的距离来集散其相似度。本文运用预先相似度计算方法来计算留言主题与相关工作部门的答复意见之间的相似度。

其中，余弦函数在三角形中的计算公式为：

$$\cos(\theta) = \frac{a^2 + b^2 + c^2}{2ab} \tag{12}$$

在直角坐标系中假设向量 a 用坐标(x_1, y_1)表示，向量 b 用坐标(x_2, y_2)表示，向量 a 和向量 b 中在直角坐标中长度为 a=$\sqrt{{x_1}^2 + {y_1}^2}$, b=$\sqrt{{x_2}^2 + {y_2}^2}$，向量 a 与 b 之间的距离我们用向量 c 表示，则 c=$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$，最后，将 a，b，c 代入三角函数的公式中得到如下的公式：

$$\begin{aligned}
\cos(\theta) &= \frac{a^2 + b^2 + c^2}{2ab} \\
&= \frac{{x_1}^2 + {y_1}^2 + {x_2}^2 + {y_2}^2 - (x_2 - x_1)^2 + (y_2 - y_1)^2}{2\sqrt{{x_1}^2 + {y_1}^2} * \sqrt{{x_2}^2 + {y_2}^2}} \\
&= \frac{x_{1*}x_2 + y_1 * y_2}{\sqrt{{x_1}^2 + {y_1}^2} \times \sqrt{{x_2}^2 + {y_2}^2}} \\
&= \frac{\sum_{i=1}^{n}(x_i + y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2}} \times \frac{1}{\sqrt{\sum_{i=1}^{n}(y_i)^2}}
\end{aligned} \tag{13}$$

其中，一个向量空间中两个向量的夹角余弦值作为衡量两个个体之间差异的大

小，当与相知接近 1，夹角趋于 0 度时，说明两个向量越相似。当余弦值接近于 0，夹角区域 90 度时，表明两个向量越不相似。

比如对已经进行分词后的两个句子：

句子 A：A 市特殊岗位

句子 B：L 市扶贫特岗

1、分词后分别得到两个列表：

ListA={‘A 市’，‘特殊’，‘岗位’}

ListB={‘A 市’，‘扶贫’，‘持’，‘岗’}

2、列出所有的词，将 ListA 和 ListB 放在一个 set 中，得到：

set={‘A 市’，‘特殊’，’ 扶贫’，‘岗位’，‘持’，‘岗’}

将上述 set 转换为 dict，key 为 set 中的词，value 为 set 中词出现的位置，即‘A 市’：1 这样的字典形式。

Dict1={‘A 市’:0，‘特殊’：1，’ 扶贫’：2，‘岗位’：3，‘持’：4，‘岗’：5 }可以看出‘A 市’这个词在 set 中排在第一，下标为 0。

3、将 ListA 于 ListB 进行编码，将每个字转换为出现在 set 中的位置，转换后为：

ListAcode={0，1，3}

ListBcode={0，2，3，4}

对于 ListAcode 于 ListBcode，可以得到 0 对应‘A 市’，3 对应‘岗位’，即 ListAcode 与 ListBcode 转换为用数字表示

4、对 ListAcode 与 ListBcode 进行 oneHot 编码，即计算每个分词出现的次数。oneHot 编号后得到的结果如下：

ListAcodeoneHot={0,1,1,1}

ListAcodeoneHot={1,0,1,1}

5、得出俩个句子的词频向量后，就变成了计算两个向量夹角的余弦值，余弦值越大相似度越高。

$$cos(\theta) = \frac{0*1+1*0+1*1+1*1}{\sqrt{(0*0+1*1+1*1+1*1)}} * \frac{1}{\sqrt{(1*1+0*0+1*1+1*1)}} = \frac{2}{3} \quad (14)$$

根据余弦值相似度，可以得出句子 A 与句子 B 相似度较高。

答复意见与留言主题相关度越高，则该答复建议对主题的价值越大，其质量越高。本文选取一个阈值，进而筛选出每个主题相关度大于该阈值的评论作为该主题下质量较高的答复建议。得到部门留言与答复意见之间的相关性关系如图 16 所示：

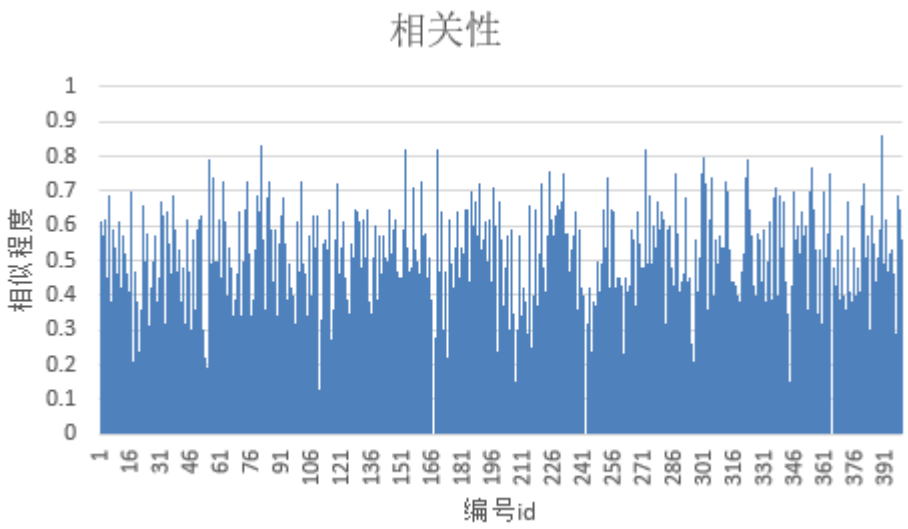

图 16 答复意见相关性

3.3.4.2 可解释性

可解释性本文指的是可以追踪到数据来源，对于答复建议而言，可解释性对于中文答复意见来说意义不大，我们可以将可解释性理解为可读性，
相关工作部门答复意见的可读性可以用自动化可读性指数 ARI（Automated Readability Index） 来表示。ARI 的计算公式为：

API=4.71*（总字符数/总字数）+0.5*（总字数/总句数）-21.43 （15）
其数值近似等于我们可以理解一段文字的最低程度。我们将绘制部分部门 id 留言以及相关部门答复意见之间的可解释性程度如图 17

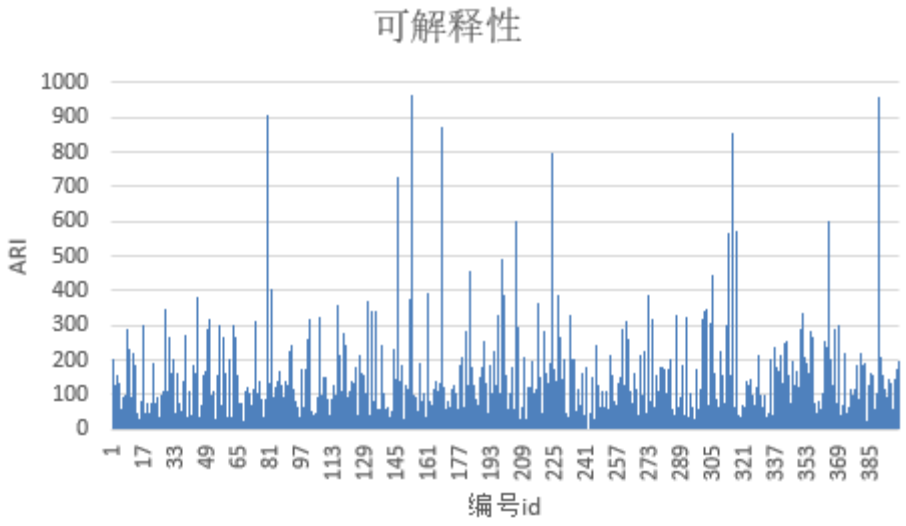

图 17 答复意见可解释性

3.3.4.3 信息量

信息量，也称答复意见长度，他表示答复意见内容的多少。通常认为，答复意见越多表明包含的有用信息越多，对于人们的参考价值越大，同时在一定程度上会增加人们对相关工作部门的信服力，以帮助留言群众可到较为满意的答复。在本文中，我们使用文本的数字表示，答复意见中内容中，少于 10 个子为 0.1 分。11 至 20 为 0.2 分，以此类推，大于 90 及以上为 1 分。

3.3.5 确定评价指标权重

关于相关工作部门答复意见质量评价模型的研究，不同的评价指标权重将会得到不同的结果。基于以上我们得到的相关评价指标，并通过计算以及编程实现得到具体的成分值，于是本文应用基于主成分分析之权值计算方法，算出为接下来即将要构建的评价模型中各个评价指标的权重。

首先将所得相关工作部门答复意见中每天数据对应的各个评价指标的数据进行标准化，以降低各个不同评价指标中的量纲差异度。本文应用 SPSS 软件自带的数据标准版方法对数据进行标准化处理。

其次将标准化后的数据导入 SPSS，对各个评价指标进行主成分分析以及权值的计算，得到结果如图 18 所示：

总方差解释

		初始特征值[a]			提取载荷平方和		
	成分	总计	方差百分比	累积 %	总计	方差百分比	累积 %
原始	1	2.209	55.216	55.216	2.209	55.216	55.216
	2	.999	24.963	80.180	.999	24.963	80.180
	3	.793	19.815	99.995	.793	19.815	99.995
	4	.000	.005	100.000	.000	.005	100.000
重新标度	1	2.209	55.216	55.216	2.209	55.216	55.216
	2	.999	24.963	80.180	.999	24.963	80.180
	3	.793	19.815	99.995	.793	19.815	99.995
	4	.000	.005	100.000	.000	.005	100.000

提取方法：主成分分析法。

a. 在分析协方差矩阵时，原始解与重新标度的解的初始特征值相同。

图 18 方差解释图

从图 18 中可直观看出，4 个主成分累计的方差贡献率超过 80%，因此 4 个主成分基本可以反应全部指标的信息。其中主成分 1 为信息量，主成分 2 为可解释性，

主成分 3 为相关性，主成分 4 为时效性。

再者，利用 SPSS 对评价指标主成分分析得到的成分矩阵如图 19 所示：

成分矩阵[a]

	原始成分				重新标度成分			
	1	2	3	4	1	2	3	4
Zscore(LEN)	.970	-.025	-.242	.010	.970	-.025	-.242	.010
Zscore(SIM)	.569	-.007	.822	.000	.569	-.007	.822	.000
Zscore(ARI)	.970	-.025	-.240	-.010	.970	-.025	-.240	-.010
Zscore(REPLY)	.052	.999	-.007	.000	.052	.999	-.007	.000

提取方法：主成分分析法。

a. 提取了 4 个成分。

图 19 成分矩阵图

基于图 18 与图 19，我们可以对信息量、可解释性、相关性、时效性 4 个主成分评价指标进行权重计算，权重确定具体计算过程如下：

首先将主成分分析中得出的“成分矩阵”及特征根输入；

然后计算线性组合中的系数，公式为：

$$b_{ij} = \frac{a_{ij}}{\sqrt{c_i}} \tag{16}$$

其中，b_j表示的是第 i 主成分第 j 变量的线性组合系数，a_{ij}表示的是第 i 主成分的第 j 变量的载荷数，c_i表示的是第 i 主成分的特征根；

进而计算综合得分模型中的系数，公式为：

$$e_j = \frac{\sum_{j=1,i=1}^{4} d_i * b_{ij}}{\sum_{i=1}^{4} d_i} \tag{17}$$

其中 d_i表示的第 i 主成分的方差，e_j表示的是得分模型中第 j 变量的系数；

最后将所有指标数据进行归一化，使其权重综合为 1，其中计算公式为：

$$index_j = \frac{e_j}{\sum_{i=1}^{4} e_i} \tag{18}$$

其中$index_j$，表示的是指标权重。

基于以上步骤，最终得到的各个不同评价指标的权重值和其模型系数如图 3 所示：

表 3 评级指标权重值以及系数表

评价指标	综合得分模型中的系数	指标权重
LEN	0.420458342	0.28
SIM	0.396046336	0.26
ARI	0.420013309	0.28
REPLY	0.270383824	0.18

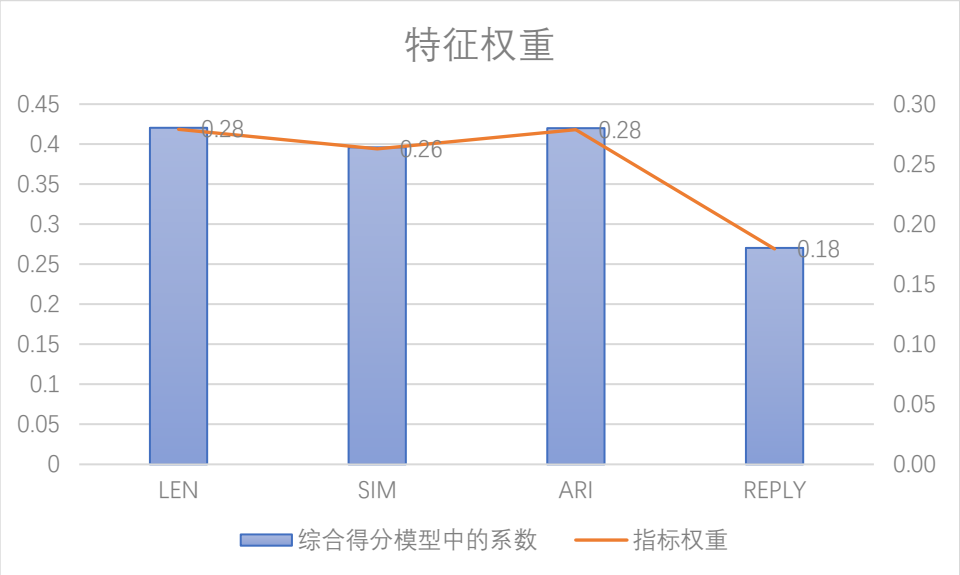

图 20 评级指标权重值以及系数图

在图 20 中，LEN、SIM、ARI、REPLY 分别对应评价指标信息量、相关性、可解释性、时效性。且其在综合得分模型中的系数分别为 0.420458342、0.396046336、0.420013309、0.270383824，其权重分别为 0.28、0.26、0.28、0.18。

3.3.6 构建答复意见质量评价指标体系

表 4 答复意见质量评价指标

指标	说明
相关性	答复意见于留言主题的相关性

可解释性	关工作部门答复意见的可读性可以用自动化可读性指数 ARI
完整性	完整的数据标记为 1，不完整则记为 0
时效性	即留言时间与答复意见时间间隔，时间越短，时效性越高，反之则越低
信息量	从内容上确保答复意见质量，以答复意见长度衡量（词/字数统计）

根据上文提取的指标特征，我们构建广义线性回归模型对相关工作部门的答复意见质量进行分析，以答复意见中提取到的相关特征，并利用成熟的回归或分类算法建立研究模型，最后对答复意见的质量进行预测。在本文中用 Quality 表示答复意见的质量，

引入符号：

Q： 得分记为

$Words$：信息量

$Relevancy$：相关性

$Credibility$：可解释性

$Timeliness$：时效性

则，建立回归模型如下

$$Q = \varphi_1 Words + \varphi_2 Relevancy + \varphi_3 Credibility + \varphi_4 Timeliness + \varepsilon \quad (19)$$

其中，ε表示常数项，$\varphi_i = \{\varphi_1，\varphi_2，\varphi_3，\varphi_4\}$表示各个评价指标对应的权值。

根据以上步骤计算得出各个评价指标的权重为$\varphi_i = \{0.28，0.26，0.28，0.18\}$代入公式（19），并经过模型训练得到

$$Q = 0.28 Words + 0.26 Relevancy + 0.28 Credibility + 0.18 Timeliness + 0.01$$

3.3.7 模型实验结果分析

在实验过程中，为了平衡模型，对文本模型的目标值数据进行标准化，保证实验数据同负一取值在[0,1]之间，并且我们规定对答复意见质量的评价，当 Q 取值在[0,0.5)之间，表示该答复意见质量较低，当 Q 取值在[0.5,1]之间则答复意见为高质量回复。

4 结论

本文经过阅读大量文献，进一步对“智慧政务“中的文本即群众的留言数、群众关心的热点问题、以及相关工作部门的解决方案数据进行内在信息的挖掘与分析。，整个过程包括数据筛选与特征提取处理，通过聚类分析，构建评价指标与建模，建模的验证分析等，最后得出本研究提出的模型具有良好的性能。本文主要结论如下：

1. 对获得的留言数据利用基于Python的fastText原理,实现了对留言数据的分类，提升了可建模度，增加了模型的准确性。
2. 聚类分析指将物理或抽象对象的集合分组成为由类似的对象组成的多个类的分析过程，利用 k-means 文本聚类算法，能更好的把留言加入对应的话题簇，能够对热点问题进行更好的分类。
3. 在构建答复意见质量评价指标与模型中，通过提取的指标特征构建广义线性回归模型对相关部门的答复意见质量进行分析。为了平衡模型，我们对文本模型的目标值进行标准化，能够使对答复意见质量的评价更为准确。

参考文献

[1]艾楚涵,姜迪,吴建德.基于主题模型和文本相似度计算的专利推荐研究[J].信息技术,2020,44(04):65-70.

[2]王光慈,汪洋.基于 FastText 的短文本分类[J].电子设计工程,2020,28(03):98-101.

[3]王俊丰,贾晓霞,李志强.基于 K-means 算法改进的短文本聚类研究与实现[J].信息技术,2019,43(12):76-80.

[4]张弛,张贯虹.基于词向量和多特征语义距离的文本聚类算法[J].重庆科技学院学报(自然科学版),2019,21(03):69-72+77.

[5]冯勇,屈渤浩,徐红艳,王嵘冰,张永刚.融合 TF-IDF 和 LDA 的中文 FastText 短文本分类方法[J].应用科学学报,2019,37(03):378-388.

[6]郭银灵. 基于文本分析的在线评论质量评价模型研究[D].内蒙古大学,2017.

[7]王小华,徐宁,谌志群.基于共词分析的文本主题词聚类与主题发现[J].情报科学,2011,29(11):1621-1624.

[6]郭银灵. 基于文本分析的在线评论质量评价模型研究[D].内蒙古大学,2017.

[8]王小华,徐宁,谌志群.基于共词分析的文本主题词聚类与主题发现[J].情报科学,2011,29(11):1621-1624.

[9]王小华,徐宁,谌志群.基于共词分析的文本主题词聚类与主题发现[J].情报科学,2011,29(11):1621-1624.

[10]Nan Hu,Indranil Bose,Noi Sian Koh etc.Manipulation od online reviews:An analysis of ratings,readability,and sentiments,2012(52):674-684.

[11]R.J.Senter,E.A.SmithAutomated readability index[OL]. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=AD06672731967.Technical Report,1997.

[12]Titov I, McDonald R. Modeling Online Reviews with Multi-grain Topic Models(C].1n: Proceedings of the17th International Conference on World Wide Web(WWW(8).New York:ACM, 2008：Il1．120.

附录

附录 1 问题 1

1.1 预测数据并计算评估值 F1

```
import os
import codecs
import xlrd
import fasttext
classifier = fasttext.load_model('../../model/model.bin')
cate_dic = {'交通运输': '交通运输', '劳动和社会保障': '劳动和社会保障', '卫生计生
': '卫生计生', '商贸旅游': '商贸旅游', '城乡建设': '城乡建设', '教育文体':'教育文
体', '环境保护':'环境保护'}
dict_cate = dict(('__label__{}'.format(v),k) for k,v in cate_dic.items())
#实际标签
label_true= []
#预测标签
label_pred = []
with open('../../data/problem1/text_data.txt','r',encoding='utf-8') as f:
    for line in f.readlines():
        line = line.strip()
        splits = line.split(" ")
        label = splits[0]
        words = [" ".join(splits[1:])]
        label = dict_cate[label]
        label_true.append(label)
        label_pred_results = classifier.predict(words)[0][0][0]
        label_pred.append(dict_cate[label_pred_results])
```

```
text_labels = list(set(label_true))
text_predict_labels = list(set(label_pred))

A = dict.fromkeys(text_labels,0)  #预测正确的各个类的数目
B = dict.fromkeys(text_labels,0)   #测试数据集中各个类的数目
C = dict.fromkeys(text_predict_labels,0) #预测结果中各个类的数目
for i in range(0,len(label_true)):
    B[label_true[i]] += 1
    C[label_pred[i]] += 1
    if label_true[i] == label_pred[i]:
        A[label_true[i]] += 1

#计算准确率 p，召回率 r，F 值
F1 = 0
for key in B:
    try:
        r = float(A[key]) / float(B[key])
        p = float(A[key]) / float(C[key])
        f = p * r * 2 / (p + r)
        F1+=f
        print("%s:\t p:%f\t r:%f\t f:%f" % (key,p,r,f))
    except:
        print("error:", key, "right:", A.get(key,0), "real:",
B.get(key,0), "predict:",C.get(key,0))

print('F1='+str(F1/len(dict_cate)))
```

1.2 构造 fasttext 模型

```
# -*- coding: UTF-8 -*-
import jieba
import fasttext

# 构建训练模型
def createClassifier():
    classifier =
fasttext.train_supervised('../../data/problem1/train_data.txt',
                                        label_prefix="__label__",
                                        dim = 150,
                                        lr = 0.2,
                                        epoch=20,
                                        word_ngrams=2,
                                        bucket= 2000000)
    classifier.save_model('../model/model.bin')
```

```
createClassifier()
```

1.3 构造停用词

```
# -*- coding: utf-8 -*-
import os
import jieba
import xlrd

#分词保存
def participle():
    data = xlrd.open_workbook("../../data/problem1/附件 2.xlsx")
    data = data.sheet_by_index(0)
    data_left = data.col_values(5)
    data_left = data_left[1: ]
    data_right = data.col_values(4)
    data_right = data_right[1: ]

    #统计词频
    TF = {}
    for filetext, label in zip(data_right, data_left):
        filetext = filetext.replace('\n', '').replace('\r',
'').replace('\t', '').replace(' ', '')
        data = ' '.join(jieba.cut(filetext, cut_all=False))  # 精确模式
        data = data.split(' ')
        if label not in TF:
            TF[label] = {}
        for x in data:
            if x in TF[label]:
                TF[label][x] += 1
            else:
                TF[label][x] = 1

    #存储频繁出现前 200 的分词
    for label, tf in TF.items():
        out = open("../../TF/"+label+".txt", 'w', encoding='utf-8')
        list = [(key, value) for key, value in tf.items()]
        list = sorted(list, key=lambda x: x[1], reverse=True)
        for x, y in list[:200]:
            out.write(x + "   " + str(y) + "\n")


#统计词频
def textData(str, words):
```

```
    f = open('../../TF/'+str, 'r', encoding= 'utf-8')
    word = (f.readline()).replace('\n', '').replace('\r',
'').replace('\t', '')
    while word:
        word = word.split(' ')[0]
        if word in words:
            words[word] += 1
        else:
            words[word] = 1
        word = (f.readline()).replace('\n', '').replace('\r',
'').replace('\t', '')
    f.close()
    return  words

#构建停用词保存
def getStopWords():
    words = {}
    words = textData('交通运输.txt', words)
    words = textData('劳动和社会保障.txt', words)
    words = textData('卫生计生.txt', words)
    words = textData('商贸旅游.txt', words)
    words = textData('城乡建设.txt', words)
    words = textData('教育文体.txt', words)
    words = textData('环境保护.txt', words)

    out = open("../../data/problem1/stopwords.txt", 'w', encoding='utf-
8')
    #当分词在超过 4 个一级标签内都存在则判定其为停用词
    for key, value in words.items():
        if value >= 4:
            out.write(key + "\n")
    out.close()

participle()
getStopWords()
```

1.4 预处理训练数据

```
# -*- coding: UTF-8 -*-
import os
import codecs
import xlrd
import jieba
import random

# 数据预处理
```

```
def init():
    #获取停用词
    stopwords = [w.strip() for w in
codecs.open("../../data/problem1/stopwords.txt", "r","utf-
8").readlines()]

    #获取语料
    data = xlrd.open_workbook("../../data/附件 2.xlsx")
    data = data.sheet_by_index(0)
    data_left = data.col_values(5)
    data_right = data.col_values(4)

    # 转换为 fasttext 规定格式
    trainSet = []
    testSet = []
    for filetext, label in zip(data_right, data_left):
        #获取一个 0 到 1 的随机浮点型数字
        rand = random.uniform(0,1)

        filetext = filetext.replace('\n', ' ').replace('\r',
'').replace('\t', '')
        #用空格分开 jieba 得到的分词
        filetext = ' '.join(w for w in jieba.cut(filetext) if w not in
stopwords)
        #加上标签名
        filetext = '__label__' + label + filetext
        #随机抽取 10%的数据为测试数据，其余为训练数据
        if rand > 0.10:
            trainSet.append(filetext)
        else:
            testSet.append(filetext)

    # 写入 train_data.txt 中
    print("writing trainSet to fasttext format...")
    out = open("../../data/problem1/train_data.txt", 'w', encoding='utf-
8')
    for sentence in trainSet[1:]:
        out.write(sentence+"\n")
    print("done!")
    out.close()

    # 写入 text_data.txt 中
    print("writing testSet to text...")
```

```
    out = open("../../data/problem1/text_data.txt", 'w', encoding='utf-
8')
    for sentence in testSet:
        out.write(sentence+'\n')
    print('done!')
    out.close()

init()
```

1.5 预测处理测试数据

```
# -*- coding: UTF-8 -*-
import os
import codecs
import xlrd
import jieba

# 测试数据预处理
def getData():
    #获取停用词
    stopwords = [w.strip() for w in
codecs.open("../../data/problem1/stopwords.txt", "r","utf-
8").readlines()]

    #获取语料
    data = xlrd.open_workbook("../../data/测试数据.xlsx")
    data = data.sheet_by_index(0)
    data_left = data.col_values(5)
    data_right = data.col_values(4)

    #转换为 fasttext 规定格式
    testSet = []
    for filetext, label in zip(data_right, data_left):
        filetext = filetext.replace('\n', ' ').replace('\r',
'').replace('\t', '')
        # 用空格分开 jieba 得到的分词
        filetext = ' '.join(w for w in jieba.cut(filetext) if w not in
stopwords)
        # 加上标签名
        filetext = '__label__' + label + filetext
        testSet.append(filetext)

    #写入 text_data.txt 中
    print("writing testSet to text...")
    out = open("../../data/problem1/text_data.txt", 'w', encoding='utf-
8')
```

```
    for sentence in testSet:
        out.write(sentence+'\n')
    print('done!')
    out.close()

getData()
```

附录二 问题 2

2.1 数据预处理

```
# -*- coding: utf-8 -*-
import pandas as pd
import jieba
import codecs
def init():
    #获取停用词
    stopwords = [w.strip() for w in codecs.open("../../data/problem2/stopwords.txt",
"r","utf-8").readlines()]

    sheet = pd.read_excel(io = '../../data/附件 3.xlsx')
    data1 = list(sheet['留言主题'])
    data2 = list(sheet['留言详情'])
    data = [x+" "+x+" "+x + " "+y for x, y in zip(data1, data2)]
    # 写入 train_data.txt 中
    print("writing trainSet to fasttext format...")
    out = open("../../data/problem2/train_data.txt", 'w', encoding='utf-8')
    for filetext in data:
        filetext = filetext.replace('\n', '').replace('\r', '').replace('\t',
'').replace(' ', '')
        filetext = ' '.join(w for w in jieba.cut_for_search(filetext) if w not in
stopwords)
        out.write(filetext + "\n")
    print("done!")
    out.close()
init()
```

2.2 获取各分类热度值

```
# -*- coding: utf-8 -*-
import pandas as pd

#计算热度值
sheet = pd.read_excel(io = '../../data/problem2/table3.xlsx')
d2 = [{"问题 ID":None,"热度指数":None,"时间范围":None}]
```

```
data = pd.DataFrame(d2)
for i in range(0, len(sheet)):
    firstTime = sheet['起始时间'][i]
    lastTime = sheet['结束时间'][i]
    label = sheet['类别'][i]
    speech_num = sheet['留言数'][i]
    agree = sheet['点赞数'][i]
    reject = sheet['反对数'][i]
    hotdays = int((str(lastTime-firstTime)).split(' ')[0])
    hot = 0.3*agree-0.1*reject+1.2*speech_num+0.8*hotdays

    data.loc[i] = {"问题 ID":label,"热度指数":hot,"时间范围":str(firstTime)+"-
"+str(lastTime)}
    print(i, agree, reject, speech_num, hotdays)

with pd.ExcelWriter(r'../../data/problem2/table1.xlsx') as writer:
    data.to_excel(writer, sheet_name='sheet')
```

2.3 统计各分类音响因子

```
# -*- coding: utf-8 -*-
import pandas as pd
import datetime
sheet = pd.read_excel(io = '../../data/problem2/table2.xlsx')
d2 = [{"类别":None,"点赞数":None,"反对数":None, '起始时间': None, '结束时间': None, '留
言数' : None}]
data = pd.DataFrame(d2)
x = 0
for i in range(0, len(sheet)):
    if isinstance(sheet['留言时间'][i], datetime.datetime):
        x = sheet['留言时间'][i].strftime('%Y-%m-%d')
        x = datetime.date(*map(int, (x.split(' ')[0]).split('-')))
        sheet.loc[i, '留言时间'] = x
    else:
        x = datetime.date(*map(int, (sheet['留言时间'][i].split(' ')[0]).split('/')))
        sheet.loc[i, '留言时间'] = x

x = 0
for j in range(0, 400):
    leibie = j
    dianzanshu = 0
    fanduishu = 0
    qishishujian = datetime.date.today()
    jieshushijian = datetime.date(2000, 1, 1)
```

```
    liuyanshu = 0
    for i in range(x, len(sheet)):
        if sheet['问题 ID'][i] == j:
            dianzanshu += sheet['点赞数'][i]
            fanduishu += sheet['反对数'][i]
            if qishishujian > sheet['留言时间'][i]:
                qishishujian = sheet['留言时间'][i]
            if jieshushijian < sheet['留言时间'][i]:
                jieshushijian = sheet['留言时间'][i]
            liuyanshu+=1
            x+=1

    data.loc[j] = {"类别":leibie,"点赞数":dianzanshu,"反对数":fanduishu, '起始时间':
qishishujian, '结束时间': jieshushijian, '留言数' : liuyanshu}
    j+=1

with pd.ExcelWriter(r'../../data/problem2/table3.xlsx') as writer:
    data.to_excel(writer, sheet_name='sheet')
```

2.4 构造停用词

```
# -*- coding: utf-8 -*-
import pandas as pd
import jieba
import codecs

def getStopwords():
    #加载附件 3
    sheet = pd.read_excel(io = '../../data/附件 3.xlsx')

    #合并主题留言与留言详情内容
    data1 = list(sheet['留言主题'])
    data2 = list(sheet['留言详情'])
    data = []
    for i in range(0, len(data1)):
        data.append(data1[i]+data2[i])

    #利用 jieba 分词并结算分词结果在各训练样本中的频率
    TF = {}
    tf = {}
    for filetext in data:
        filetext = filetext.replace('\n', '').replace('\r', '').replace('\t',
'').replace(' ', '')
        temp = jieba.cut_for_search(filetext)
```

```
        for x in temp:
            if x not in tf:
                tf[x] = 1
        for key, value in tf.items():

            if key not in TF:
                TF[key] = 1
            else:
                TF[key] += 1
        tf = {}
    #将词频排序
    fs = ((x, y) for x, y in TF.items())
    fs = sorted(fs, key=lambda x: x[1], reverse=True)

    #将频率低于 3 归为停用词
    print("writing testSet to text...")
    out = open("../../data/problem2/stopwords.txt", 'w', encoding='utf-8')
    for x, y in fs:
        if  y < 3:
            out.write(x + "\n")
    out.close()
getStopwords()
```

2.5 获取各分类关键词

```
import pandas as pd
import jieba
import codecs

#得到的训练数据
set = pd.read_excel(io = '../../data/problem2/table2.xlsx')
#获取停用词
stopwords = [w.strip() for w in codecs.open("../../data/problem2/stopwords.txt",
"r","utf-8").readlines()]

data1 = list(set['留言主题'])
data2 = list(set['留言详情'])
data = [x + y for x, y in zip(data1, data2)]

for i in range(0, len(data)):
    filetext = data[i].replace('\n', '').replace('\r', '').replace('\t', '').replace('
', '')
    filetext = [w for w in jieba.cut_for_search(filetext) if w not in stopwords]
    data[i] = filetext
```

```
#计算每个分类频率排行前 10 的分词
out = open("../../data/problem2/hotTopic.txt", 'w', encoding='utf-8')
x = 0
for i in range(0, 400):
    TF = {}
    for j in range(0, len(data)):
        if set['问题 ID'][j] != i:
            continue
        for key in data[j]:
            if key in TF:
                TF[key] += 1
            else:
                TF[key] = 1
    fs = ((x, y) for x, y in TF.items())
    fs = sorted(fs, key=lambda x: x[1], reverse=True)
    j = 0
    filetext = " "
    for x, y in fs:
        if j > 10:
            break
        filetext += x
        filetext += " "
        j+=1
    out.write("第"+str(i)+"类关键词："+filetext + "\n")
    print("第"+str(i)+"类关键词："+filetext + "\n")
    i+=1
out.close()
```

2.6 Kmeans 聚类

```
# coding=utf-8
import time
import re
import os
import sys
import codecs
import shutil
import numpy as np
import pandas as pd
import openpyxl
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt

# 文档预料 空格连接
corpus = []
# 读取预料 一行预料为一个文档
for line in open('../../data/problem2/train_data.txt', 'r',
encoding='utf8').readlines():
    corpus.append(line.strip())
# print(corpus)
# 将文本中的词语转换为词频矩阵 矩阵元素 a[i][j] 表示 j 词在 i 类文本下的词频
vectorizer = CountVectorizer()
# 该类会统计每个词语的 tf-idf 权值
transformer = TfidfTransformer()

# 第一个 fit_transform 是计算 tf-idf 第二个 fit_transform 是将文本转为词频矩阵
tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))

# 获取词袋模型中的所有词语
word = vectorizer.get_feature_names()

# 将 tf-idf 矩阵抽取出来，元素 w[i][j]表示 j 词在 i 类文本中的 tf-idf 权重
weight = tfidf.toarray()
# 打印特征向量文本内容
resName = "../../data/problem2/tfidf.txt"
result = codecs.open(resName, 'w', 'utf-8')
for j in range(len(word)):
    result.write(word[j] + ' ')
result.write('\r\n\r\n')

# 打印每类文本的 tf-idf 词语权重，第一个 for 遍历所有文本，第二个 for 便利某一类文本下的词
语权重
for i in range(len(weight)):
    for j in range(len(word)):
        # print weight[i][j],
        result.write(str(weight[i][j]) + ' ')
    result.write('\r\n\r\n')

result.close()
print('.................Kmeans.................')
clf = KMeans(n_clusters=400,init='random',n_init=10)
clf.fit(weight)
print(clf.inertia_)
sheet = pd.read_excel(io = '../../data/附件 3.xlsx')
sheet['问题 ID'] = None
```

```
i = 0
while i <= len(clf.labels_):
    print(i)
    sheet.loc[i-1,'问题 ID'] = clf.labels_[i - 1]
    i+=1
sheet = sheet.sort_values(by = '问题 ID')

with pd.ExcelWriter(r'../../data/problem2/table2.xlsx') as writer:
    sheet.to_excel(writer, sheet_name='sheet3')
```

附录三 问题 3

3.1 答复评估

```
import jieba
import math
import codecs
import pandas as pd
import string
import datetime
def aCountable(dataSet):
    data = list(dataSet['答复意见'])
    punc = string.punctuation
    count=0·
    ARI = []
    for filetext in data:
        filetext = filetext.replace('\n', '').replace('\r', '').replace('\t',
'').replace(' ', '')
        cnt = 0         #总数
        isTrue = 0      #文字
        isFalse = 0     #符号
        completec = 0  #完整句
        for word in filetext:
            if word not in punc:
                isTrue+=1
            else:
                if word in ['？','。','！']:
                    completec+=1
                isFalse+=1
            cnt+=1
        ari = 4.71*(isFalse/cnt)+0.5*(isTrue/(completec+1))-21.43
        count+=1
        ARI.append(ari)
    return ARI

def relevance(dataSet):
```

```
    # 载入停用词
    stopwords = [w.strip() for w in codecs.open("../../data/problem3/stopwords.txt",
"r", "utf-8").readlines()]

    # 载入数据集
    data1 = list(dataSet['留言主题'])
    data2 = list(dataSet['留言详情'])
    data = [x + " " + y for x, y in zip(data1, data2)]
    data3 = list(dataSet['答复意见'])

    _sim = 0  # 总相似度
    _cnt = 0  # 总文本数
    SIM = []
    sim = {'x <= 20%': 0, '20% < x  <= 40%': 0, '40% <  x <= 60%': 0, 'x > 60%': 0}  #
相似度分布
    cnt = {'x <= 20%': 0, '20% < x  <= 40%': 0, '40% <  x <= 60%': 0, 'x > 60%': 0}  #
文本数分布
    for query, answer in zip(data, data3):
        # 分词
        query = query.replace('\n', '').replace('\r', '').replace('\t', '').replace('
', '')
        query = [w for w in jieba.cut_for_search(query) if w not in stopwords]

        answer = answer.replace('\n', '').replace('\r', '').replace('\t', '').replace('
', '')
        answer = [w for w in jieba.cut_for_search(answer) if w not in stopwords]

        # 取分词交集去重
        word_set = set(query).union(set(answer))

        # 建立分词与数值的映射关系
        word_dict = dict()
        i = 0
        for word in word_set:
            word_dict[word] = i
            i += 1

        query_cut_code = [word_dict[word] for word in query]
        query_cut_code = [0] * len(word_dict)

        for word in query:
            query_cut_code[word_dict[word]] += 1

        answer_cut_code = [word_dict[word] for word in answer]
```

```
        answer_cut_code = [0] * len(word_dict)
        for word in answer:
            answer_cut_code[word_dict[word]] += 1

        # 计算余弦相似度
        sum = 0
        sq1 = 0
        sq2 = 0
        for i in range(len(query_cut_code)):
            sum += query_cut_code[i] * answer_cut_code[i]
            sq1 += pow(query_cut_code[i], 2)
            sq2 += pow(answer_cut_code[i], 2)

        try:
            result = round(float(sum) / (math.sqrt(sq1) * math.sqrt(sq2)), 2)
        except ZeroDivisionError:
            result = 0.0
        SIM.append(result)
        if result <= 0.2:
            sim['x <= 20%'] += result
            cnt['x <= 20%'] += 1
        elif result <= 0.4:
            sim['20% < x  <= 40%'] += result
            cnt['20% < x  <= 40%'] += 1
        elif result <= 0.6:
            sim['40% <  x <= 60%'] += result
            cnt['40% <  x <= 60%'] += 1
        else:
            sim['x > 60%'] += result
            cnt['x > 60%'] += 1
        _sim += result
        _cnt += 1
    print('       x <= 20%: ' + str(sim['x <= 20%'] / cnt['x <= 20%']) + " 共计：" +
str(cnt['x <= 20%']))
    print('20% < x  <= 40%: ' + str(sim['20% < x  <= 40%'] / cnt['20% < x  <= 40%']) +
" 共计：" + str(
        cnt['20% < x  <= 40%']))
    print('40% <  x <= 60%: ' + str(sim['40% <  x <= 60%'] / cnt['40% <  x <= 60%']) +
" 共计：" + str(
        cnt['40% <  x <= 60%']))
    print('       x > 60%: ' + str(sim['x > 60%'] / cnt['x > 60%']) + " 共计：" +
str(cnt['x > 60%']))
    print(_sim / _cnt)
    return SIM
```

```
def length(dataSet):
    data = dataSet['答复意见']
    LEN = []
    CPL = []
    for filetext in data:
        LEN.append(len(filetext))
        if len(filetext)==0:
            CPL.append(0)
        else:
            CPL.append(1)
    return LEN,CPL

def reply(dataSet):
    dataF = dataSet['留言时间']
    dataL = dataSet['答复时间']
    REPLY = []
    for first, last in zip(dataF, dataL):
        if isinstance(first, datetime.datetime):
            first = first.strftime('%Y-%m-%d')
            first = datetime.date(*map(int, (first.split(' ')[0]).split('-')))
        else:
            first = datetime.date(*map(int, (first.split(' ')[0]).split('/')))

        if isinstance(last, datetime.datetime):
            last = last.strftime('%Y-%m-%d')
            last = datetime.date(*map(int, (last.split(' ')[0]).split('-')))
        else:
            last = datetime.date(*map(int, (last.split(' ')[0]).split('/')))

        REPLY.append(last-first)
    return REPLY

dataSet = pd.read_excel(io='../../data/附件 4.xlsx')
LEN,CPL = length(dataSet)
SIM = relevance(dataSet)
ARI = aCountable(dataSet)
REPLY = reply(dataSet)
set = {'LEN':LEN, 'SIM':SIM,'ARI':ARI,'CPL':CPL, 'REPLY':REPLY}

df = pd.DataFrame(set)
print(set)
with pd.ExcelWriter(r'../../data/problem3/table.xlsx') as writer:
    df.to_excel(writer, sheet_name='sheet')
```