第八届泰迪杯数据挖掘竞赛 C 题

“智慧政务”中的文本挖掘应用

Text Mining Application in "Smart Government Affairs"

摘要

近年来网络问政平台逐步成为政府了解民意、 汇聚民智、凝聚民气的重要渠道，各类社情民意相关的文本数据量不断攀升，给以往主要依 靠人工来进行留言划分和热点整理的相关部门的工作带来了极大挑战。同时，随着大数据、 云计算、人工智能等技术的发展，建立基于自然语言处理技术的智慧政务系统已经是社会治理创新发展的新趋势，对提升政府的管理水平和施政效率具有极大的推动作用。本文运用 XGboost 算法对政府网站留言进行了自动分类，并利用 LDA 模型对各个留言进行了文本挖掘将其按主题归类，并进行了热点追踪，结果表明利用大数据来辅助处理网民留言是十分有效的。

关键词：XGboost；LDA 主题模型；文本挖掘；机器学习；智慧政务；

Abstract

In recent years, the Internet questioning platform has gradually become an important channel for the government to understand public opinion, gather people's wisdom, and gather popular sentiment. The amount of text data related to various social conditions and public opinions has been increasing. This has brought great challenges to the work of the relevant departments that rely on manual to divide the message and sort out the hot issues. At the same time, with the development of big data, cloud computing, artificial intelligence and other technologies, the establishment of a smart government system based on natural language processing technology has become a new trend of social governance innovation and development, which has a great impact on improving the government's management level and governance efficiency. It will greatly promote the management level and governance efficiency of the government. This article uses the XGboost algorithm to automatically classify messages on government websites, and uses the LDA model to perform text mining on each message to categorize it by topic and track hotspots. The results show that it is Very effective to use big data to assist in processing netizen messages of.

Key words: XGboost; LDA theme model; text mining; machine learning; smart government;

目录

引言

随着人工智能的发展，自然语言处理（NLP）逐渐成为一个重要的研究方向。近些年来，在社交网络的非常快速的兴起过程中，微博、哔哩哔哩、市长网箱等由网络新媒体、互联网信息公开平台组成的网络政府平台已经逐步成为政府了解民意、 汇聚民智、凝聚民气的重要渠道，众所周知，互联网是一个具有开放性、全天性以及匿名性的人工系统，因此网络民意留言的信息数量将会是非常巨大，并且因为其匿名性，时常会有许多无意义的垃圾留言充斥其中，对相关民意处理部门的工作会带来巨大的挑战

随着大数据、云计算以及机器学习和人工智能技术的不断发展，基于网络留言这类自然语言的文本信息挖掘处理技术已经成熟，并且已经有很多学者发表做过相关研究，也涌现了非常多的相关挖掘与处理工具，相关的模型建立以及检验算法也不断完善，诸如 F-Score 检验、LDA 主题模型、交叉验证等以及非常的成熟，但对于政府网络信息公开和留言分析相关的技术研究却鲜有提及

综上所述，本队选择 C 题进行解答，运用 XGboost 以及 MBUT-LDA 对留言文本进行了分类与聚类以及热点问题的主题追踪，并对模型做出了检验。

1 挖掘流程和原理分析

1.1 总体流程图

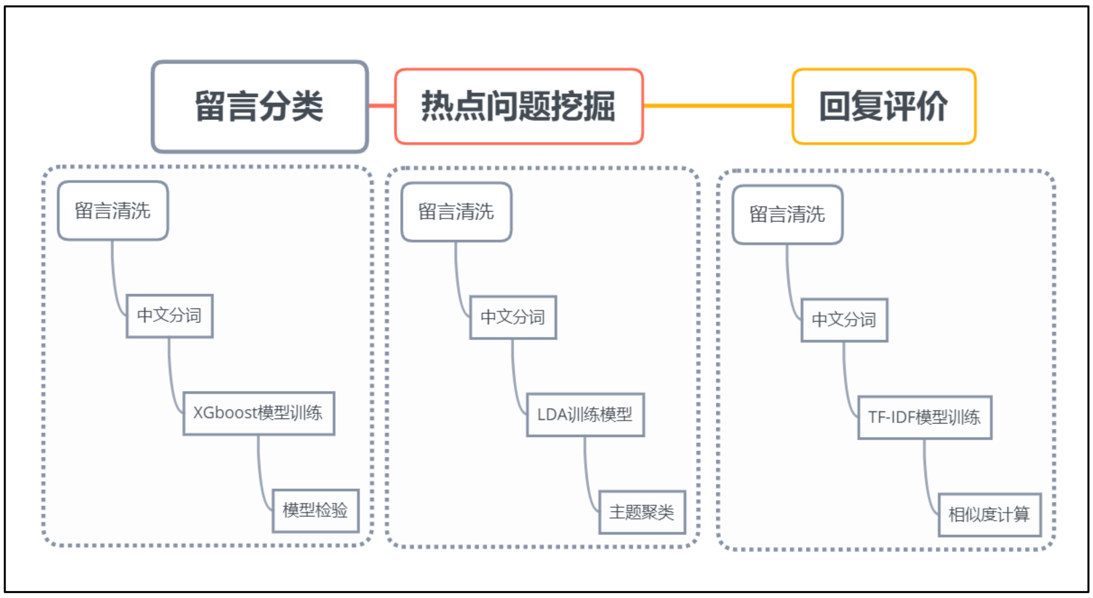


图 1-1 流程图

1.2 原理分析

1.2.1 数据清洗

在众多留言数据中，出现了很多重复的数据。考虑部分极端网络用户会大量发送垃圾留言或者灌水留言，因此在去重的时候应该取更新时间最晚的一条。考虑到在内存中大量缓存数据会影响程序性能，因此在清洗数据时候，仅读取需要的列，并删除重复行，单独写入一个新的清洗后的数据文件，以此提供后续的运行效率。

1.2.2 JIEBA 中文分词

我们在对留言信息进行挖掘分析之前，先要把非结构化的文本信息转换为计算机能够识别的结构化信息。在众多留言中，为了便于转换，先要对这些留言信息进行中文分词。这里采用 Python 的中文分词包 JIEBA 进行分词。JIEBA 采用了采用了动态规划查找最大概率路径，找出基于词频的最大切分组合，这里采用 JIEBA 自带的语义库，并添加了部分政务相关的词库提高后续 NPL 自然语言处理提高准确率。

1.2.3 TF-IDF

在对留言信息分词后，需要把这些词语转换为向量，以供挖掘分析使用。这里采用 TF-IDF 算法，把留言信息转换为词袋向量[1]。TF-IDF 算法的具体原理如下：

第一步，计算词频，即 TF 权重（Term Frequency）。

$$TF = \text{某个词在文本中出现的次数}$$

考虑到留言有长短之分，为了便于不同留言的比较，进行“词频“标准化，除以文本的总词数或者除以该文本中出现次数最多的词的出现次数即：

$$TF = \frac{\text{某个词出现的次数}}{\text{文本的总词数}}$$

第二步，计算 IDF 权重，即逆文档频率（Inverse Document Frequency），需要建立一个语料库（corpus），用来模拟语言的使用环境。IDF 越大，此特征性在文本中的分布越集中，说明该分词在区分该文本内容属性能力越强：

$$IDF = log\,(\frac{\text{语料库的文本总数}}{\text{保护该词的文本数} + 1})$$

第三步，计算 TF-IDF 值（Term Frequency Document Frequency）：

$$TF - IDF = TF \times IDF$$

实际分析得出 TF-IDF 值与一个词在职位描述表中文本出现的次数成正比，某个词文本的重要性越高，TF-IDF 值越大[2]。计算文本中每个词的 TF-IDF 值，进行排序，次数最多的即为要提取的职位描述表中文本的关键词。

1.2.4 生成 TF-IDF 词袋向量

生成 TF-IDF 向量的具体步骤如下：

（1）使用 TF-IDF 算法，找出每个职位描述的前 5 个关键词；

（2）对每个岗位描述提取的 5 个关键词，合并成一个集合，计算每个岗位描述对于这个集合中词的词频，如果没有则记为 0；

（3）生成各个岗位描述的 TF-IDF 权重向量，计算公式如下：

$$TF - IDF = TF \times IDF$$

1.2.5 XGboost 多分类

要对留言就行分类就需要用到回归树，XGboost 用到了 CART 回归树，其主要原理如下：

第一步，对分类对象经过一系降维列化简得到：

$$obj^{*} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_i+\lambda}+\gamma T$$

第二步，进一步降维可以得到如下式子：

$$g_i = \partial_{\hat{y}(t-1)} l(y_i, \hat{y}^{(t-1)}), h_i = \partial\partial^2_{\hat{y}(t-1)}\, l(y_i, \hat{y}^{(t-1)})$$

联立得：

$$G_j = \sum_{i\in I_j} g_i\,, H_j = \sum_{i\in I_j} h_i$$

这个对象（obj）代表了每棵树的要优化的最终目标，应该越小越好，我们前面得到了 t-1 个函数模型，我们要构建第 t 个模型时，只要求前 t-1 函数的一阶和二阶导数和前 t-1 的函数模型的乘积即可[3]，这里的可以控制叶子节点的个数，继而控制模型复杂度，提高范化能力，其公式如下：

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}+\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right]-\gamma$$

1.2.6 LDA 主题模型

对文本聚类就需要提取主题，用到主题模型，本题中我对使用的是 LDA 主题模型，其主要原理如下：

在 LDA 中，所有的描述相当于是词袋，每个袋子里都有一堆词，用的时候就只管检测这些词出现与否就，用公式可以表示成：

$$P\left(主题\middle|特征，语料库\right) = \frac{此特征出现的次数}{文本包含的所有的特征} \times 该文本属于该主题的特征数$$

利用朴素贝叶斯模型计算该特征的概率：

$$P\left(主题\middle|语料库\right) = \frac{P(主题)P(特征|语料库)}{\sum P\left(主题\right)P(特征|语料库)}$$

经过一系列化简，可以得到这样一个关系：

$$P(词|文本) = P（词|主题）P（主题|文本）$$

同一主题下，某个词出现的概率，以及同一留言下，某个主题出现的概率，两个概率的乘积，可以得到某篇文本出现某个词的概率，我们在训练的时候，调整这两个分布就可以得到更加精确的主题模型，综上所述，其主要原理即[4]：

$$pw_i = \frac{主题t对应到V中第i个单词出现的次数}{主题t下的所有单词总数}$$

1.2.7 TF-IDF 相关性和完整性评价

如果回复内容和主题的相关性为 0，那么我们就认为这条回复不相关，再对比所有的回复，回复的相关性即为：

$$总体相关性 = \frac{相关性不为0的回复数}{回复总数} \times 100\%$$

获取回复中的主要的内容，以此为回复的标准格式，再将回复的标准格式和回复的内容进行比较，获得内容的相似度，如果回复内容完整性低于某一个阈值，就认为这条回复是不完整的，再对比所有的回复，回复的完整性即为：

$$完整性 = \frac{\sum(回复完整性)}{总体回复条数}$$

这里 k 即为完整性阈值，在进行多次拟合后，选取动态平均值作为阈值，得出的所给数据的总体完整性公式：

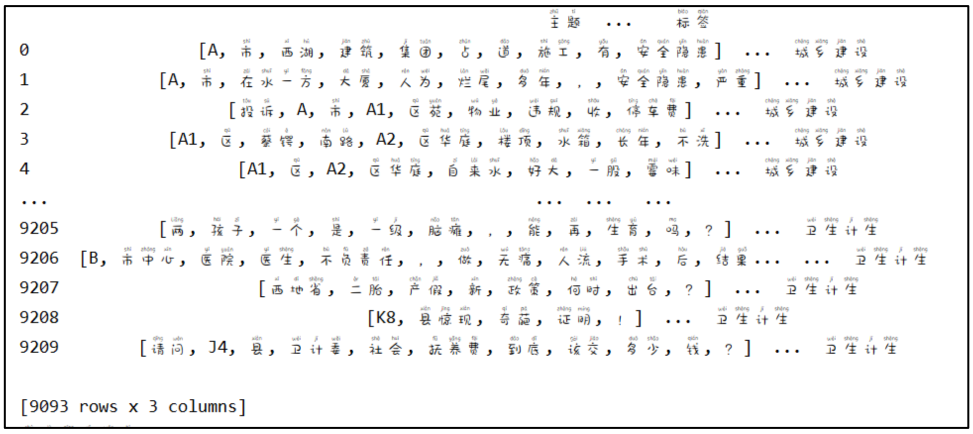
$$总体完整性 = \frac{\sum\left(回复完整性 > \widehat{完整性}\right)}{总体回复条数} \times 100\%$$

2 实验过程及方法

2.1 利用 XGboost 对留言进行标签分类

2.1.1 对留言数据进行清洗并 Jieba 分词

首先对所有留言数据进行文本清洗，利用留言内容的比较将重复的垃圾留言剔除，并根据中文日常用语词根库以及政务词库利用 Jieba 分词模块对留言的主题和内容进行分词，并用拉丁化符号、西文符号以及中文诸如“的、请、您”等语气虚词设置停止词，剔除这些无意义的词汇，对原始的留言数据进行数据清洗以及分词操作。



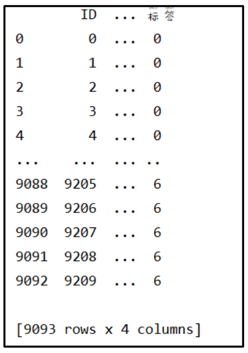
```
                                              主题  ...    标签
0              [A, 市, 西湖, 建筑, 集团, 占, 道, 施工, 有, 安全隐患]  ...  城乡建设
1          [A, 市, 在水一方, 大厦, 人为, 烂尾, 多年, , , 安全隐患, 严重]  ...  城乡建设
2                  [投诉, A, 市, A1, 区苑, 物业, 违规, 收, 停车费]  ...  城乡建设
3          [A1, 区, 蔡锷, 南路, A2, 区华庭, 楼顶, 水箱, 长年, 不洗]  ...  城乡建设
4                  [A1, 区, A2, 区华庭, 自来水, 好大, 一股, 霉味]  ...  城乡建设
...                                            ...  ...   ...
9205       [两, 孩子, 一个, 是, 一级, 脑瘫, , , 能, 再, 生育, 吗, ?]  ...  卫生计生
9206  [B, 市中心, 医院, 医生, 不负责任, , , 做, 无痛, 人流, 手术, 后, 结果...  ...  卫生计生
9207                 [西地省, 二胎, 产假, 新, 政策, 何时, 出台, ?]  ...  卫生计生
9208                         [K8, 县惊现, 奇葩, 证明, !]  ...  卫生计生
9209     [请问, J4, 县, 卫计委, 社会, 抚养费, 到底, 该交, 多少, 钱, ?]  ...  卫生计生

[9093 rows x 3 columns]
```

图 2-1-1 清洗后的数据部分分词展示

2.1.2 将分词留言矩阵化并建立模型

为方便模型的建立，将清洗后的分词数据的标签数字规划，用 int 型 0-6 的数字分别替代之前诸如“城乡建设”、“环境保护”等标签，便于之后模型的建立，用规划后的标签作为检验的 Y 项，随机取原数据的部分作为训练集，另一部分作为测试集。

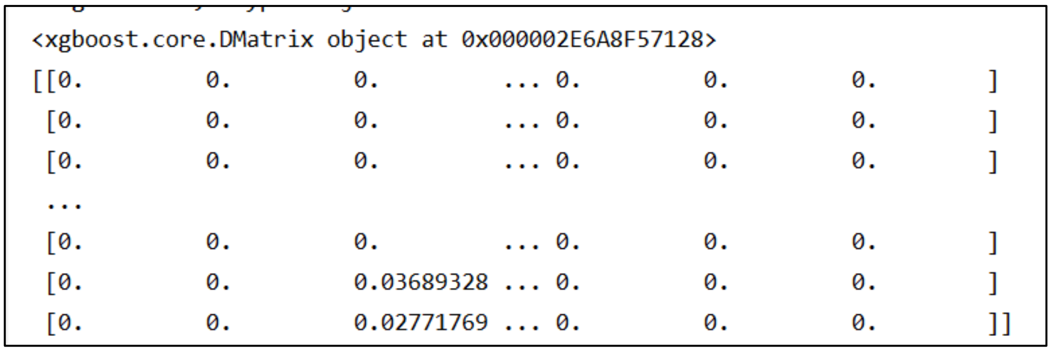


```
        ID  ... 标签
0        0  ...  0
1        1  ...  0
2        2  ...  0
3        3  ...  0
4        4  ...  0
...    ...  ... ..
9088  9205  ...  6
9089  9206  ...  6
9090  9207  ...  6
9091  9208  ...  6
9092  9209  ...  6

[9093 rows x 4 columns]
```

图 2-1-2 规划后的标签展示

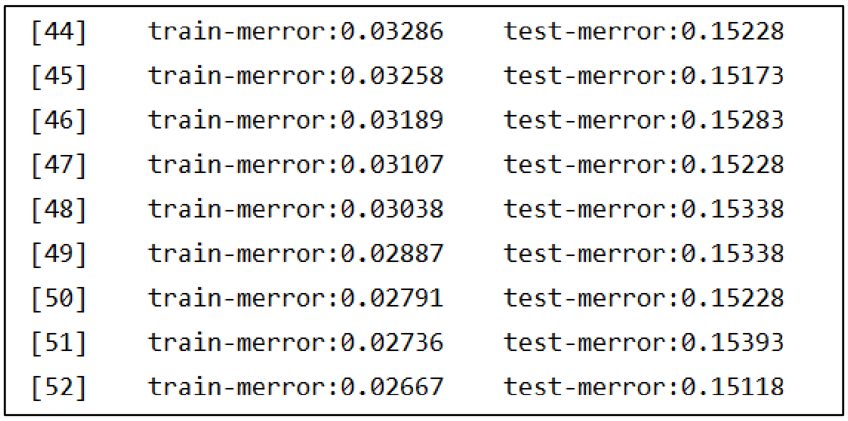
利用词频统计建立字典，将分词的语料数据规划为词袋向量，将词袋向量作为模型标签的变量，在实际操作中，因为留言会分为主题和内容，根据以往经验，主题往往是所反应诉求或内容的概况，因此主题是对留言问题分类来说是十分重要的，但也有非常多的留言会在内容中概述与主题无关的内容，但却又直指主要诉求，因此单独以主题或者内容作为模型训练参数都会导致模型的偏差，因此我们给主题和内容分别设置了不同的权重，以更真实的反应留言的诉求类型。



```
<xgboost.core.DMatrix object at 0x000002E6A8F57128>
[[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.         0.03689328 ... 0.         0.         0.        ]
 [0.         0.         0.02771769 ... 0.         0.         0.        ]]
```

图 2-1-3 部分规划后的词袋矩阵

规划完词袋矩阵后，利用 XGboost 解析语料参数，并将其分为 7 类，利用当前数据的一部分作为训练集，对训练集参数解析后进行标签打分，并同原始标签进行对比，并进行模型修正，反复迭代 500 次后即可以得到相应的分类模型。

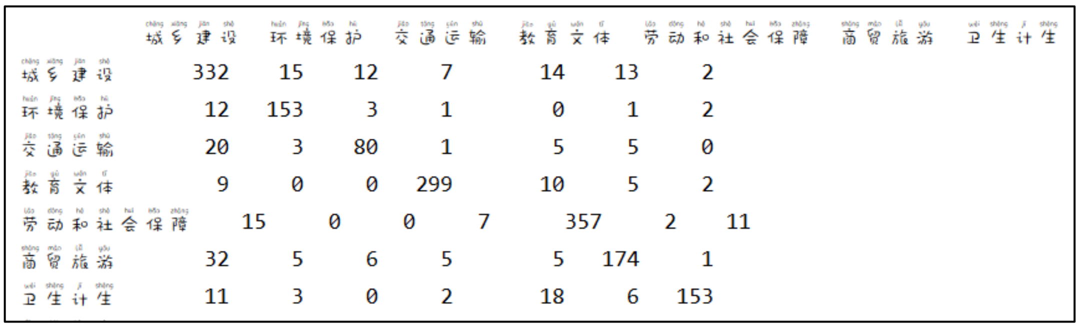


```
[44]	train-merror:0.03286	test-merror:0.15228
[45]	train-merror:0.03258	test-merror:0.15173
[46]	train-merror:0.03189	test-merror:0.15283
[47]	train-merror:0.03107	test-merror:0.15228
[48]	train-merror:0.03038	test-merror:0.15338
[49]	train-merror:0.02887	test-merror:0.15338
[50]	train-merror:0.02791	test-merror:0.15228
[51]	train-merror:0.02736	test-merror:0.15393
[52]	train-merror:0.02667	test-merror:0.15118
```

图 2-1-4 部分迭代权重展示

2.1.3 模型检验

为了验证模型的准确性，这里引入了交叉验证，即利用一部分数据作为测试数据，用之前训练的模型对传入的分词语料进行标签打分，并同真实的数据进行交叉检验



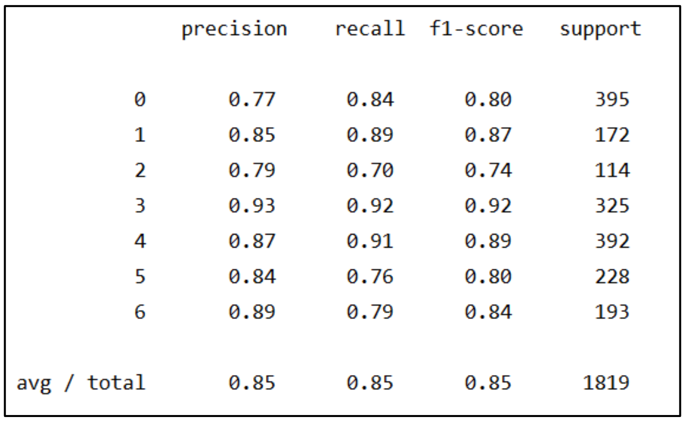
	城乡建设	环境保护	交通运输	教育文体	劳动和社会保障	商贸旅游	卫生计生
城乡建设	332	15	12	7	14	13	2
环境保护	12	153	3	1	0	1	2
交通运输	20	3	80	1	5	5	0
教育文体	9	0	0	299	10	5	2
劳动和社会保障	15	0	0	7	357	2	11
商贸旅游	32	5	6	5	5	174	1
卫生计生	11	3	0	2	18	6	153

图 2-1-5 交叉验证结果

根据如下公式，利用 F-Score 对所得到的模型进行结果检验

$$F_1 = \frac{1}{n}\sum_{i=1}^{1}\frac{2RR_i}{Pi + R_i}$$

得到的验证结果如下：



```
             precision    recall  f1-score   support

          0       0.77      0.84      0.80       395
          1       0.85      0.89      0.87       172
          2       0.79      0.70      0.74       114
          3       0.93      0.92      0.92       325
          4       0.87      0.91      0.89       392
          5       0.84      0.76      0.80       228
          6       0.89      0.79      0.84       193

avg / total       0.85      0.85      0.85      1819
```

图 2-1-6 模型 F 检验

得到 F1 测度为：

0.94289726187690l5

模型准确率高达:

90.15170423309511 %

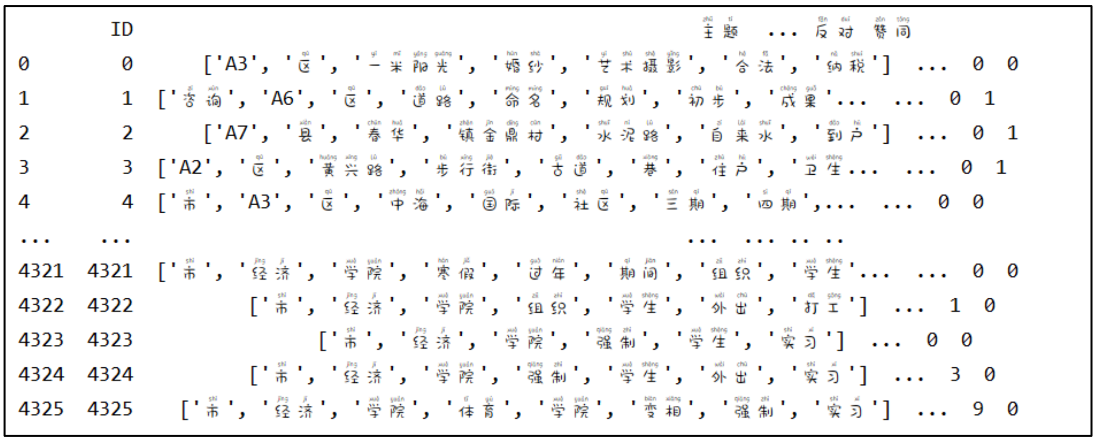
2.2 利用 LDA 主题模型进行热点挖掘

某一问题可能在一给时段内集中反映，这些问题很可能是亟待解决的重难点，因此提取留言中的主要内容，特别是与地点、人群有关的名词、客观词在一个时间序列里集中出现既可以称为是热点问题，在这里本对将留言语料规划为字典，

训练出 LDA 主题模型，并根据点赞和反对数配置权重，进行时间序列分析，即可得出热点问题[5]。

2.2.1 对留言进行分词清理

对所有留言数据进行文本清洗，将重复的留言剔除，根据日常用语及政务词库利用 Jieba 分词模块对留言的主题进行分词，并将拉丁化符号、西文符号以及中文虚词剔除，对原始的留言数据进行数据清洗以及分词操作。

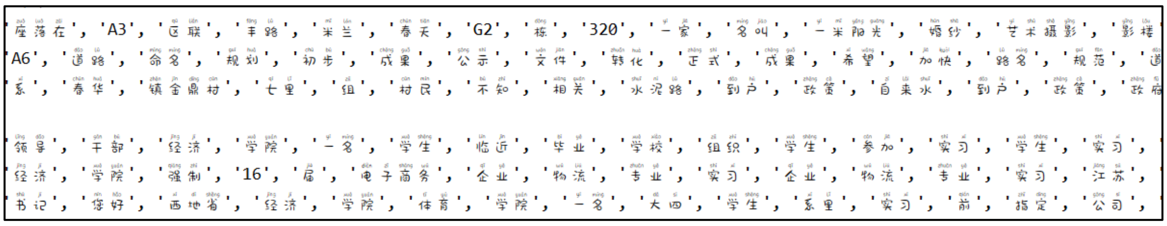


```
          ID                                         主题  ... 反对 赞同
0         0      ['A3', '区', '一米阳光', '婚纱', '艺术摄影', '合法', '纳税']  ...  0  0
1         1  ['咨询', 'A6', '区', '道路', '命名', '规划', '初步', '成果'...  ...  0  1
2         2      ['A7', '县', '春华', '镇金鼎村', '水泥路', '自来水', '到户']  ...  0  1
3         3  ['A2', '区', '黄兴路', '步行街', '古道', '巷', '住户', '卫生...  ...  0  1
4         4  ['市', 'A3', '区', '中海', '国际', '社区', '三期', '四期',...  ...  0  0
...     ...                                          ...  ...  ..  ..
4321   4321  ['市', '经济', '学院', '寒假', '过年', '期间', '组织', '学生'...  ...  0  0
4322   4322          ['市', '经济', '学院', '组织', '学生', '外出', '打工']  ...  1  0
4323   4323                ['市', '经济', '学院', '强制', '学生', '实习']  ...  0  0
4324   4324          ['市', '经济', '学院', '强制', '学生', '外出', '实习']  ...  3  0
4325   4325    ['市', '经济', '学院', '体育', '学院', '变相', '强制', '实习']  ...  9  0
```

图 2-2-1 清洗后的数据部分展示

2.2.2 LDA 主题模型建立

利用清洗后的数据构建留言语料库，统计词频量，并以此构建字典，利用 Gensim 可以辅助完成这些操作，在字典的基础上建立分词引索，并将其规划为 TF-IDF 词袋向量，此时词袋向量既可以初略反应留言文本的特征信息：

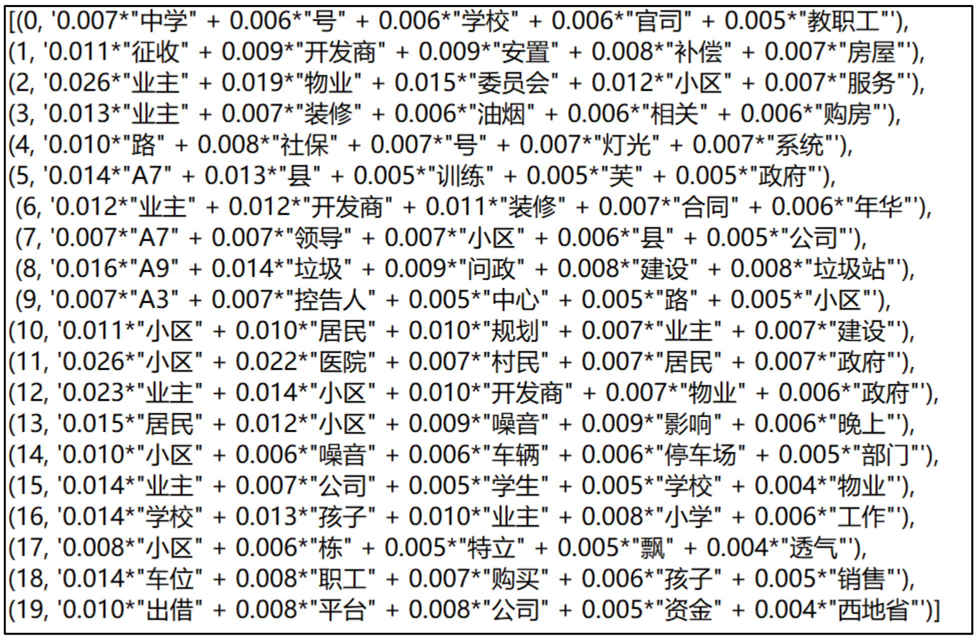


```
'座落在', 'A3', '区联', '丰路', '米兰', '春天', 'G2', '栋', '320', '一家', '名叫', '一米阳光', '婚纱', '艺术摄影', '影楼
'A6', '道路', '命名', '规划', '初步', '成果', '公示', '文件', '转化', '正式', '成果', '希望', '加快', '路名', '规范', '道
'系', '春华', '镇金鼎村', '七里', '组', '村民', '不知', '相关', '水泥路', '到户', '政策', '自来水', '到户', '政策', '政府

'领导', '干部', '经济', '学院', '一名', '学生', '临近', '毕业', '学校', '组织', '学生', '参加', '实习', '学生', '实习',
'经济', '学院', '强制', '16', '届', '电子商务', '企业', '物流', '专业', '实习', '企业', '物流', '专业', '实习', '江苏',
'书记', '您好', '西地省', '经济', '学院', '体育', '学院', '一名', '大四', '学生', '系里', '实习', '前', '指定', '公司',
```

图 2-2-2 构建的语料库部分展示

利用 TF-IDF 词袋向量训练 MBUT-LDA 主题模型，并进行循环迭代检验

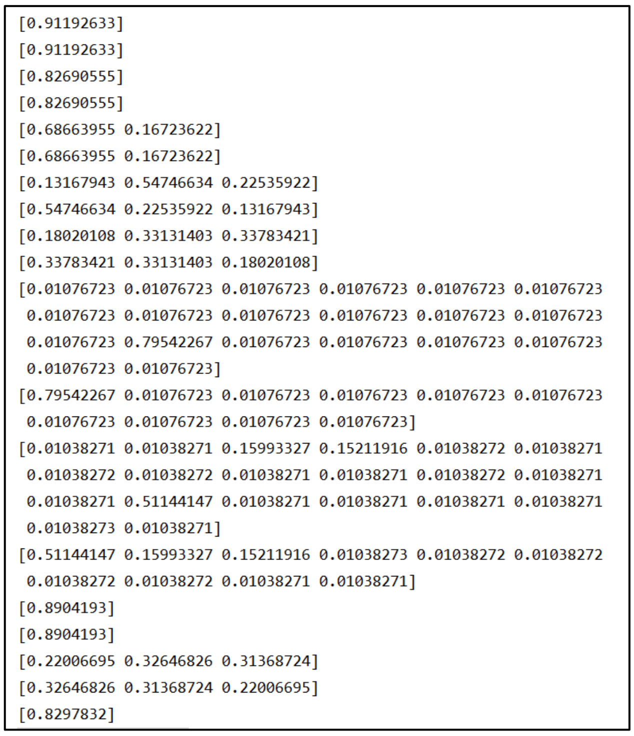


```
[(0, '0.007*"中学" + 0.006*"号" + 0.006*"学校" + 0.006*"官司" + 0.005*"教职工"'),
(1, '0.011*"征收" + 0.009*"开发商" + 0.009*"安置" + 0.008*"补偿" + 0.007*"房屋"'),
(2, '0.026*"业主" + 0.019*"物业" + 0.015*"委员会" + 0.012*"小区" + 0.007*"服务"'),
(3, '0.013*"业主" + 0.007*"装修" + 0.006*"油烟" + 0.006*"相关" + 0.006*"购房"'),
(4, '0.010*"路" + 0.008*"社保" + 0.007*"号" + 0.007*"灯光" + 0.007*"系统"'),
(5, '0.014*"A7" + 0.013*"县" + 0.005*"训练" + 0.005*"芙" + 0.005*"政府"'),
 (6, '0.012*"业主" + 0.012*"开发商" + 0.011*"装修" + 0.007*"合同" + 0.006*"年华"'),
 (7, '0.007*"A7" + 0.007*"领导" + 0.007*"小区" + 0.006*"县" + 0.005*"公司"'),
 (8, '0.016*"A9" + 0.014*"垃圾" + 0.009*"问政" + 0.008*"建设" + 0.008*"垃圾站"'),
 (9, '0.007*"A3" + 0.007*"控告人" + 0.005*"中心" + 0.005*"路" + 0.005*"小区"'),
(10, '0.011*"小区" + 0.010*"居民" + 0.010*"规划" + 0.007*"业主" + 0.007*"建设"'),
(11, '0.026*"小区" + 0.022*"医院" + 0.007*"村民" + 0.007*"居民" + 0.007*"政府"'),
(12, '0.023*"业主" + 0.014*"小区" + 0.010*"开发商" + 0.007*"物业" + 0.006*"政府"'),
(13, '0.015*"居民" + 0.012*"小区" + 0.009*"噪音" + 0.009*"影响" + 0.006*"晚上"'),
(14, '0.010*"小区" + 0.006*"噪音" + 0.006*"车辆" + 0.006*"停车场" + 0.005*"部门"'),
(15, '0.014*"业主" + 0.007*"公司" + 0.005*"学生" + 0.005*"学校" + 0.004*"物业"'),
(16, '0.014*"学校" + 0.013*"孩子" + 0.010*"业主" + 0.008*"小学" + 0.006*"工作"'),
(17, '0.008*"小区" + 0.006*"栋" + 0.005*"特立" + 0.005*"飘" + 0.004*"透气"'),
(18, '0.014*"车位" + 0.008*"职工" + 0.007*"购买" + 0.006*"孩子" + 0.005*"销售"'),
(19, '0.010*"出借" + 0.008*"平台" + 0.008*"公司" + 0.005*"资金" + 0.004*"西地省"')]
```

图 2-2-3 部分典型主题权重

2.2.3 获取热点问题

将训练得到的模型反哺原始的留言文本，获取主题分布时间序列：



```
[0.91192633]
[0.91192633]
[0.82690555]
[0.82690555]
[0.68663955 0.16723622]
[0.68663955 0.16723622]
[0.13167943 0.54746634 0.22535922]
[0.54746634 0.22535922 0.13167943]
[0.18020108 0.33131403 0.33783421]
[0.33783421 0.33131403 0.18020108]
[0.01076723 0.01076723 0.01076723 0.01076723 0.01076723 0.01076723
 0.01076723 0.01076723 0.01076723 0.01076723 0.01076723 0.01076723
 0.01076723 0.79542267 0.01076723 0.01076723 0.01076723 0.01076723
 0.01076723 0.01076723]
[0.79542267 0.01076723 0.01076723 0.01076723 0.01076723 0.01076723
 0.01076723 0.01076723 0.01076723 0.01076723]
[0.01038271 0.01038271 0.15993327 0.15211916 0.01038272 0.01038271
 0.01038272 0.01038272 0.01038271 0.01038271 0.01038272 0.01038271
 0.01038271 0.51144147 0.01038271 0.01038271 0.01038271 0.01038271
 0.01038273 0.01038271]
[0.51144147 0.15993327 0.15211916 0.01038273 0.01038272 0.01038272
 0.01038272 0.01038272 0.01038271 0.01038271]
[0.8904193]
[0.8904193]
[0.22006695 0.32646826 0.31368724]
[0.32646826 0.31368724 0.22006695]
[0.8297832]
```

图 2-2-4 可视化的主题分布

根据热度分布以及点赞的权重获取排名前五的主题：

主题排名	关键词	热度序列
1	中学 号 学校 官司 教职工 检察院 县 下区 法院 解决	[0.00674831 0.00625456 0.00612691 0.00601054 0.0052419 0.00427084 0.00382323 0.00379121 0.00355975 0.00336647]
2	征收 开发商 安置 补偿 房屋 业主 自治 号 交房 拆迁	[0.01142097 0.00913383 0.00866168 0.00789249 0.00727021 0.0062578 0.0058234 0.00573004 0.00539628 0.00514325]
3	业主 物业 委员会 小区 服务 物业管理 城市 业主大会 标准 建设	[0.02556376 0.01888852 0.01478885 0.0115629 0.00651406 0.00618977 0.00612689 0.00593402 0.00571318 0.00545192]
4	业主 装修 油烟 相关 购房 部门 销售 2018 开发商 文件	[0.01306833 0.00672773 0.00647434 0.00646785 0.0063652 0.00634834 0.00577478 0.00563298 0.0054645 0.00532184]
5	路 社保 号 灯光 系统 路口 记录 车辆 道 直行	[0.00999885 0.00791689 0.00730615 0.00717076 0.006969 0.00553872 0.00544821 0.00541748 0.00518617 0.00455785]

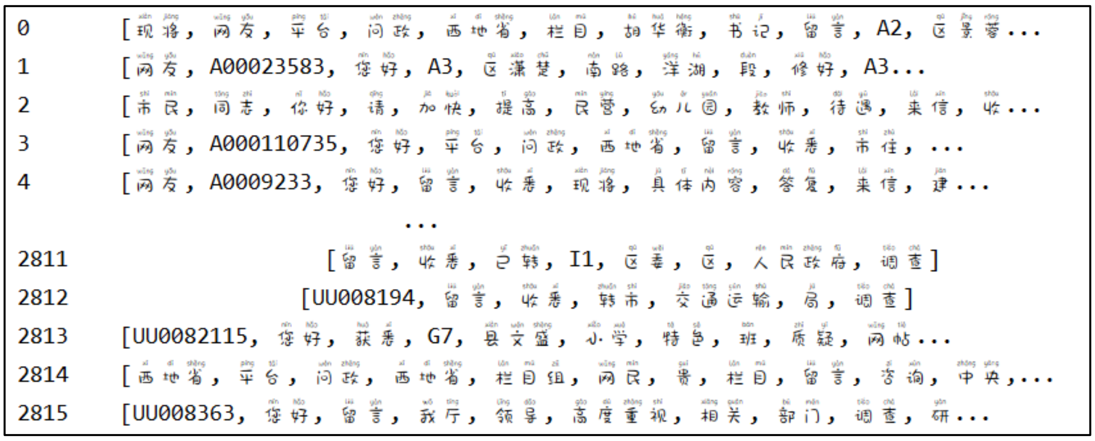
图 2-2-5 排名前五的主题

2.3 利用 TF-IDF 进行回复评价

政府对社会问题的回复常常成为普通民众关注的焦点，不合时宜或者不小心错序的回复常常引发大量网络使用者和普通民众讨论的热点话题，造成极其不好的负面影响，而利用模型在对外公布前进行回复评价，并根据评价结果对回复进行修正或发布后，可以在一定程度上规避因偶然的意外照成的回复错序等情况。

2.3.1 对回复进行分词清洗

对所有回复数据进行文本清洗，根据日常用语及政务词库利用 Jieba 分词模块对回复的进行分词，并将拉丁化符号、西文符号以及中文虚词剔除，对原始的回复数据进行数据清洗以及分词操作[6]。

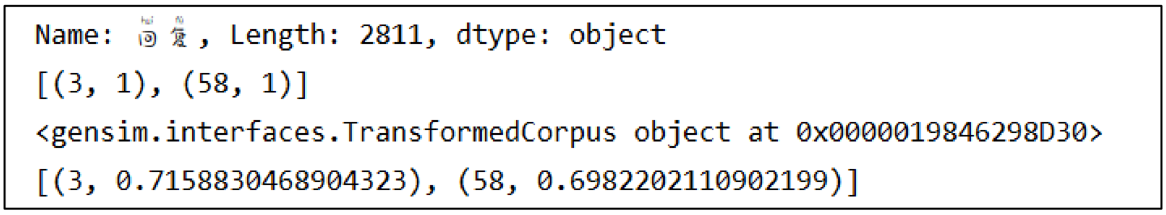


```
0       [现将, 网友, 平台, 问政, 西地省, 栏目, 胡华衡, 书记, 留言, A2, 区景蓉...
1       [网友, A00023583, 您好, A3, 区潇楚, 南路, 洋湖, 段, 修好, A3...
2       [市民, 同志, 你好, 请, 加快, 提高, 民营, 幼儿园, 教师, 待遇, 来信, 收...
3       [网友, A000110735, 您好, 平台, 问政, 西地省, 留言, 收悉, 市住, ...
4       [网友, A0009233, 您好, 留言, 收悉, 现将, 具体内容, 答复, 来信, 建...
                         ...
2811                [留言, 收悉, 已转, I1, 区委, 区, 人民政府, 调查]
2812              [UU008194, 留言, 收悉, 转市, 交通运输, 局, 调查]
2813    [UU0082115, 您好, 获悉, G7, 县文盛, 小学, 特色, 班, 质疑, 网帖...
2814    [西地省, 平台, 问政, 西地省, 栏目组, 网民, 贵, 栏目, 留言, 咨询, 中央,...
2815    [UU008363, 您好, 留言, 我厅, 领导, 高度重视, 相关, 部门, 调查, 研...
```

图 2-3-1 清洗后的回复

2.3.2 TF-IDF 模型建立

利用清洗后的数据构建回复文本集，利用文本集生成分词列表，统计词频量，基于文本集建立词典，并获得词典特征数，基于词典将分词列表集转换成稀疏向量集，构建语料库，用词典把回复词也转换为稀疏向量创建 TF-IDF 模型，传入语料库来训练模型。

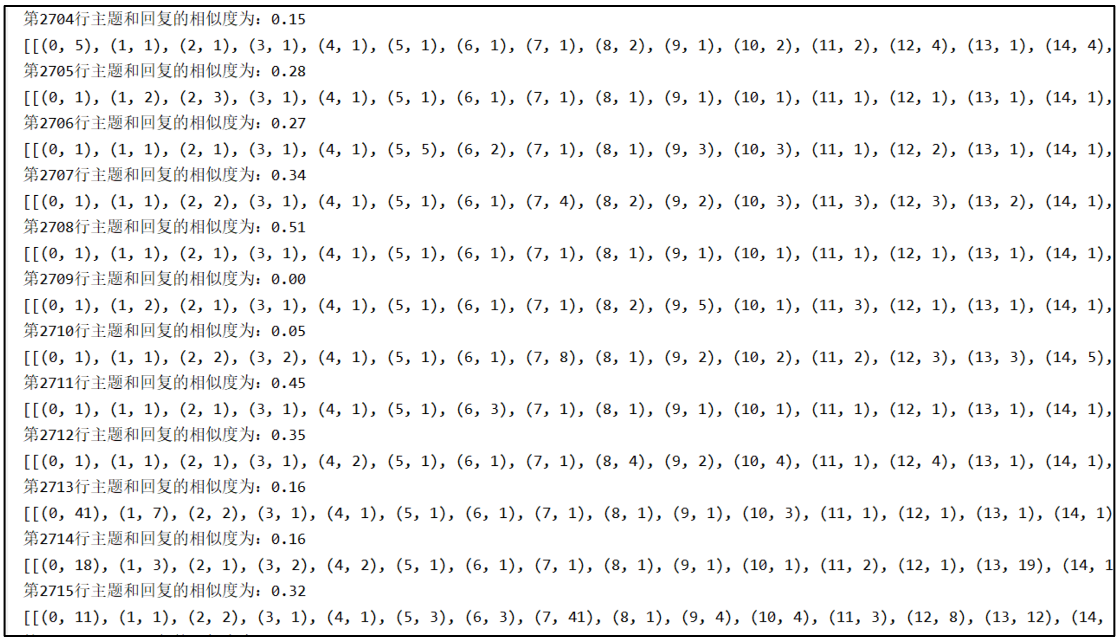


```
Name: 回复, Length: 2811, dtype: object
[(3, 1), (58, 1)]
<gensim.interfaces.TransformedCorpus object at 0x0000019846298D30>
[(3, 0.7158830468904323), (58, 0.6982202110902199)]
```

图 2-3-2 构建的模型参数

2.3.3 相关性检验

用训练好的 TF-IDF 模型处理回复的文档和原本的留言内容，这里将语料库用作参数，同回复进行 SparseMatrixSimilarity 相似度检验，以判断回复与留言的吻合度，以此对回复做出评价[7]。



```
第2704行主题和回复的相似度为: 0.15
[[(0, 5), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 2), (9, 1), (10, 2), (11, 2), (12, 4), (13, 1), (14, 4),
第2705行主题和回复的相似度为: 0.28
[[(0, 1), (1, 2), (2, 3), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1),
第2706行主题和回复的相似度为: 0.27
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 5), (6, 2), (7, 1), (8, 1), (9, 3), (10, 3), (11, 1), (12, 2), (13, 1), (14, 1),
第2707行主题和回复的相似度为: 0.34
[[(0, 1), (1, 1), (2, 2), (3, 1), (4, 1), (5, 1), (6, 1), (7, 4), (8, 2), (9, 2), (10, 3), (11, 3), (12, 3), (13, 2), (14, 1),
第2708行主题和回复的相似度为: 0.51
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1),
第2709行主题和回复的相似度为: 0.00
[[(0, 1), (1, 2), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 2), (9, 5), (10, 1), (11, 3), (12, 1), (13, 1), (14, 1),
第2710行主题和回复的相似度为: 0.05
[[(0, 1), (1, 1), (2, 2), (3, 2), (4, 1), (5, 1), (6, 1), (7, 8), (8, 1), (9, 2), (10, 2), (11, 2), (12, 3), (13, 3), (14, 5),
第2711行主题和回复的相似度为: 0.45
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 3), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1),
第2712行主题和回复的相似度为: 0.35
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 2), (5, 1), (6, 1), (7, 1), (8, 4), (9, 2), (10, 4), (11, 1), (12, 4), (13, 1), (14, 1),
第2713行主题和回复的相似度为: 0.16
[[(0, 41), (1, 7), (2, 2), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 3), (11, 1), (12, 1), (13, 1), (14, 1)
第2714行主题和回复的相似度为: 0.16
[[(0, 18), (1, 3), (2, 1), (3, 2), (4, 2), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 2), (12, 1), (13, 19), (14, 1
第2715行主题和回复的相似度为: 0.32
[[(0, 11), (1, 1), (2, 2), (3, 1), (4, 1), (5, 3), (6, 3), (7, 41), (8, 1), (9, 4), (10, 4), (11, 3), (12, 8), (13, 12), (14,
```

图 2-3-3 相似度检验部分展示

如果回复内容和主题的相关性为 0，那么我们就认为这条回复不相关，再对比所有的回复，回复的相关性即为：

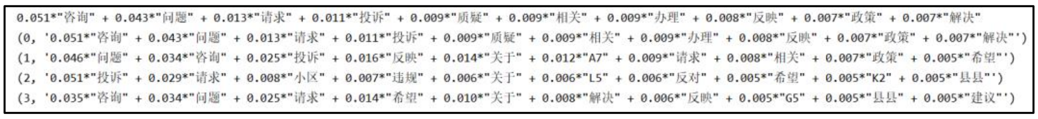
$$总体相关性 = \frac{相关性不为0\ 的回复数}{回复总数} \times 100\%$$

经过模型训练验证，所给数据总体回复相关性为：

$$93.11324103656372\%$$

2.3.4 完整性检验

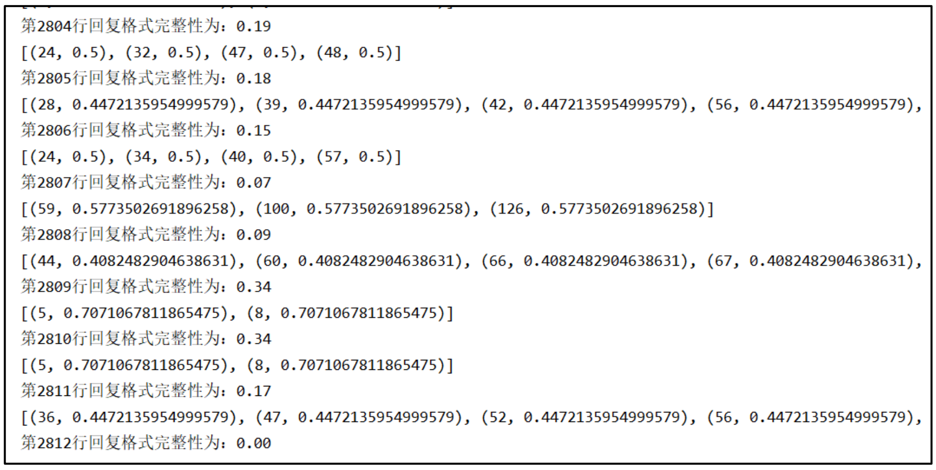
用清洗完毕的 TF-IDF 词袋向量进行 LDA 主题模型训练，可以得出留言回复的主题关键词，将其关键词同留言的主题进行交叉检验，用以判断回复留言的主题是否合适，以此对回复做出完整性评价。



```
0.051*"咨询" + 0.043*"问题" + 0.013*"请求" + 0.011*"投诉" + 0.009*"质疑" + 0.009*"相关" + 0.009*"办理" + 0.008*"反映" + 0.007*"政策" + 0.007*"解决"
(0, '0.051*"咨询" + 0.043*"问题" + 0.013*"请求" + 0.011*"投诉" + 0.009*"质疑" + 0.009*"相关" + 0.009*"办理" + 0.008*"反映" + 0.007*"政策" + 0.007*"解决"')
(1, '0.046*"问题" + 0.034*"咨询" + 0.025*"投诉" + 0.016*"反映" + 0.014*"关于" + 0.012*"A7" + 0.009*"请求" + 0.008*"相关" + 0.007*"政策" + 0.005*"希望"')
(2, '0.051*"投诉" + 0.029*"请求" + 0.008*"小区" + 0.007*"违规" + 0.006*"关于" + 0.006*"L5" + 0.006*"反对" + 0.005*"希望" + 0.005*"K2" + 0.005*"县县"')
(3, '0.035*"咨询" + 0.034*"问题" + 0.025*"请求" + 0.014*"希望" + 0.010*"关于" + 0.008*"解决" + 0.006*"反映" + 0.005*"G5" + 0.005*"县县" + 0.005*"建议"')
```

图表 2-3-4 部分完整性主题检验

获取回复中的主要的内容，以此为回复的标准格式，再将回复的标准格式和回复的内容进行比较，获得内容的相似度



```
第2804行回复格式完整性为: 0.19
[(24, 0.5), (32, 0.5), (47, 0.5), (48, 0.5)]
第2805行回复格式完整性为: 0.18
[(28, 0.4472135954999579), (39, 0.4472135954999579), (42, 0.4472135954999579), (56, 0.4472135954999579),
第2806行回复格式完整性为: 0.15
[(24, 0.5), (34, 0.5), (40, 0.5), (57, 0.5)]
第2807行回复格式完整性为: 0.07
[(59, 0.5773502691896258), (100, 0.5773502691896258), (126, 0.5773502691896258)]
第2808行回复格式完整性为: 0.09
[(44, 0.4082482904638631), (60, 0.4082482904638631), (66, 0.4082482904638631), (67, 0.4082482904638631),
第2809行回复格式完整性为: 0.34
[(5, 0.7071067811865475), (8, 0.7071067811865475)]
第2810行回复格式完整性为: 0.34
[(5, 0.7071067811865475), (8, 0.7071067811865475)]
第2811行回复格式完整性为: 0.17
[(36, 0.4472135954999579), (47, 0.4472135954999579), (52, 0.4472135954999579), (56, 0.4472135954999579),
第2812行回复格式完整性为: 0.00
```

图表 2-3-5 完整性检验

如果回复内容完整性低于某一个阈值，就认为这条回复是不完整的，再对比所有的回复，回复的完整性即为：

$$总体完整性 = \frac{\sum\left(回复完整性 - \widehat{完性整}\right)}{总体回复条数} \times 100\%$$

这里 k 即为完整性阈值，我队在进行多次拟合后，选取完整性的平均值作为完整性阈值，得出的所给数据的总体完整性为：

$$40.14909478168264\%$$

3 代码

3.1 XGboost 文本 NPL 分类

3.1.1 数据清洗

```python
import warnings

import pandas as pd
import jieba

warnings.filterwarnings('ignore')  # 取消警告
# 读取数据

data = pd.read_excel('../数据/附件 2.xlsx', usecols=[2, 4, 5])

data.columns = ['主题', '内容', '标签']

# 清除空行, 范围全部行, 是的
data.dropna(how='all', inplace=True)
# 删除重复行
data.drop_duplicates(inplace=True)

print("正在清洗中.....")

# 添加自定义词汇

jieba.load_userdict('../清洗规则/添加词汇.txt')

# 过滤规则

stopWords = pd.read_csv('../清洗规则/stopword.txt', encoding='GB18030',

sep='hahaha', header=None)

stopWords = ['≮', '≯', '≠', '≮', '会', '月', '日', '–', '\t\n', '*'] +

list(stopWords.iloc[:, 0])
# 结巴分词

data_head = data['主题'].apply(lambda x: jieba.lcut(x))

data_val = data['内容'].apply(lambda x: jieba.lcut(x))

df = pd.DataFrame({'主题': data_head, '内容': data_val, '标签': data['标签

']})
```

```
df.to_excel('../清洗后数据/附件2_第一问_清洗后.xlsx')

print(df)

print("数据清洗完毕")
```

3.1.2 XGboost 模型训练及 F 检验

```
import warnings
from sklearn import metrics

warnings.filterwarnings('ignore')  # 取消警告
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
import xgboost as xgb
import pandas as pd
import numpy as np

# 读取数据

train_data = pd.read_excel('../清洗后数据/附件2_第一问_清洗后.xlsx',

usecols=[0, 1, 2, 3], header=0)

train_data.columns = ['ID', '主题', '内容', '标签']

# 把纯数字字符串转换为 int 型

class_mapping = {'城乡建设': 0, '环境保护': 1, '交通运输': 2, '教育文体': 3, '

劳动和社会保障': 4, '商贸旅游': 5, '卫生计生': 6}

train_data['标签'] = train_data['标签'].map(class_mapping)

# 打印数据
print(train_data)
# 定义变量和标签

X = train_data['内容'] + train_data['主题']  # 1 和 2 列

Y = train_data['标签']  # 第 3 列

# 分割数据集为训练和测试
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
# 将语料转化为词袋向量，根据词袋向量统计 TF-IDF
valNums = CountVectorizer(max_features=1200)
```

```
tf_idf_transformer = TfidfTransformer()
tf_idf = tf_idf_transformer.fit_transform(valNums.fit_transform(x_train))
x_train_weight = tf_idf.toarray()  # 训练集 TF-IDF 权重矩阵
tf_idf = tf_idf_transformer.transform(valNums.transform(x_test))
x_test_weight = tf_idf.toarray()  # 测试集 TF-IDF 权重矩阵
# DMatrix 是 XGBoost 的二进制的缓存文件, 加载的数据存储在对象 DMatrix 中。
xgb_train = xgb.DMatrix(x_train_weight, label=y_train)
xgb_test = xgb.DMatrix(x_test_weight, label=y_test)
print(x_train)
print(xgb_train)
print(x_train_weight)

print("开始训练！")

# 参数
param = {'silent': 0,
        'eta': 0.3,
        'max_depth': 6,
        'objective': 'multi:softmax',
        'num_class': 7,  # 类别
        'eval_metric': 'merror'}

watchlist = [(xgb_train, 'train'), (xgb_test, 'test')]
# 循环次数
num_round = 100
# 训练模型
xgb_model = xgb.train(param, xgb_train, num_round, watchlist)
# 保存训练模型

xgb_model.save_model('../模型/XGB.model')

# 模型预测
y_predict = xgb_model.predict(xgb_test)
# 给标签

label_all = ['城乡建设', '环境保护', '交通运输', '教育文体', '劳动和社会保障',

'商贸旅游', '卫生计生']

confusion_mat = metrics.confusion_matrix(y_test, y_predict)
df = pd.DataFrame(confusion_mat, columns=label_all)
df.index = label_all

print('准确率：\n', metrics.accuracy_score(y_test, y_predict))

print('交叉验证:\n', df)
print('分类报告:\n', metrics.classification_report(y_test, y_predict))

# F1 测度
confusion_mat = confusion_mat.astype('float') /
confusion_mat.sum(axis=1)[:, np.newaxis]  # 归一化,左上角特异度, 右下角灵敏度
TP = confusion_mat[1, 1]
TN = confusion_mat[0, 0]
FP = confusion_mat[0, 1]
FN = confusion_mat[1, 0]
precision = TP / (TP + FP)
recall = TP / (TP + FN)
F1 = 2 * recall * precision / (recall + precision)

print("F1 测度 :  ", F1)
```

3.2 LDA 主题模型训练及热点问题挖掘

```
import time
import warnings

import gensim
import jieba
import numpy as np
import pandas as pd
from gensim import corpora, models

warnings.filterwarnings('ignore')  # 取消警告


print('1.读取数据 ------')

t_start = time.time()  # 即时开始

data = pd.read_excel('../数据/附件3.xlsx', usecols=[2, 3, 4, 5, 6])  # 读取
数据

data.columns = ['主题', '时间', '内容', '反对', '认同']

# 清除空行, 范围全部行, 是的
data.dropna(how='all', inplace=True)
# 删除重复行
data.drop_duplicates(inplace=True)

print('读入数据完成, 用时%.3f秒' % (time.time() - t_start))
print('2.数据清洗 ------')

t_start = time.time()  # 即时开始

data_head = data['内容'].apply(lambda x: jieba.lcut(x))  # 初始化 jieba

stopWords = pd.read_csv('../清洗规则/stopword.txt', encoding='GB18030',

sep='hahaha', header=None)  # 过滤规则

stopWords = [' ', '全', '年', '中', '请', '会', '月', '市', '区', 'A', '日',

'万', '说', '-', '\t', '\n', '\u3000', '\xa0', '\r\n',

            '“', '”'] + list(stopWords.iloc[:, 0])
data_head = data_head.apply(lambda x: [ic for ic in x if ic not in
stopWords])  # 应用规则

print('数据清洗完成，用时%.3f 秒' % (time.time() - t_start))

print('3.构建语料库 ------')

t_start = time.time()  # 即时开始
doclist = data_head.values  # 获取语料
print(doclist)  # 显示语料
stoplist = ['']  # 停止词
# 读取语料
texts = [[word for word in doc if word not in stoplist] for doc in doclist]

print('语料库构建完成，用时%.3f 秒' % (time.time() - t_start))

print('4.构建字典 ------')

t_start = time.time()  # 即时开始
dictionary = corpora.Dictionary(texts)  # 获取字典
corpus = [dictionary.doc2bow(text) for text in texts]  # 获取引索
print(corpus[13])  # 显示

print('字典构建完成，用时%.3f 秒' % (time.time() - t_start))

print('5.构建文本向量 ------')

t_start = time.time()  # 即时开始
M = len(texts)
# 计算 tf-idf 值
corpus_tfidf = models.TfidfModel(corpus)[corpus]

print('建立向量完成，用时%.3f 秒' % (time.time() - t_start))
print('6.LDA 训练模型 ------')

t_start = time.time()  # 即时开始
lda = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary,
num_topics=20)
lda.print_topic(10, topn=5)  # 显示前 5 行

lda.save('../模型/LDA.model')  # 保存模型

print(lda.print_topics(num_topics=20, num_words=5))  # 显示前 20,5 个词语

print('LDA 模型完成，用时%.3f 秒' % (time.time() - t_start))


print('7.详细结果主题分布：--')

t_start = time.time()  # 即时开始
doc_topics = lda.get_document_topics(corpus_tfidf)  # 所有文档的主题分布
idx = np.arange(M)
np.random.shuffle(idx)
idx = idx[:10]
num_show_topic = 10  # 每个文档显示前几个主题
for i in idx:
    topic = np.array(doc_topics[i])
    topic_distribute = np.array(topic[:, 1])
    print(topic_distribute)  # 显示详情
    topic_idx = topic_distribute.argsort()[:-num_show_topic - 1:-1]
    print(topic_distribute[topic_idx])

print('8.结果：前主题的词分布：--')

num_show_term = 20  # 每个主题显示几个词
num_topics = 10  # 显示主题
for topic_id in range(num_topics):

    print('主题#%d：\t' % topic_id)

    term_distribute_all = lda.get_topic_terms(topicid=topic_id)
    term_distribute = term_distribute_all[:num_show_term]
    term_distribute = np.array(term_distribute)
    term_id = term_distribute[:, 0].astype(np.int)

    print('词：\t', )

    for t in term_id:
        print(dictionary.id2token[t], )

    print('\n概率：\t', term_distribute[:, 1])
print('用时%.3f秒' % (time.time() - t_start))
```

3.3 TF-IDF 进行回复评价

3.3.1 TF-IDF 相关性评价

```
import jieba
import xlrd
from gensim.corpora import Dictionary
from gensim.models import TfidfModel
from gensim.similarities import SparseMatrixSimilarity


data = xlrd.open_workbook('../数据/附件4.xlsx')

st = data.sheets()[0]
nrows = st.nrows
st1 = st.col_slice(2, start_rowx=0, end_rowx=None)
st2 = st.col_slice(5, start_rowx=0, end_rowx=None)
count = 0.0
for i in range(1, nrows):
    a = st1[i].value
    b = st2[i].value
    texts = [b, '']
    keyword = a
    # 1、将【文本集】生成【分词列表】
    texts = [jieba.lcut(text) for text in texts]
    # 2、基于文本集建立【词典】，并获得词典特征数
    dictionary = Dictionary(texts)
    num_features = len(dictionary.token2id)
    # 3.1、基于词典，将【分词列表集】转换成【稀疏向量集】，称作【语料库】
    corpus = [dictionary.doc2bow(text) for text in texts]
    # 3.2、同理，用【词典】把【搜索词】也转换为【稀疏向量】
    kw_vector = dictionary.doc2bow(jieba.lcut(keyword))
    # 4、创建【TF-IDF 模型】，传入【语料库】来训练
    tfidf = TfidfModel(corpus)

    tfidf.save('../模型/TF-IDF.model')  # 保存模型

    # 5、用训练好的【TF-IDF 模型】处理【被检索文本】和【搜索词】
    tf_texts = tfidf[corpus]  # 此处将【语料库】用作【被检索文本】
    tf_kw = tfidf[kw_vector]
    # 6、相似度计算
    sparse_matrix = SparseMatrixSimilarity(tf_texts, num_features)
    similarities = sparse_matrix.get_similarities(tf_kw)
    s = similarities[0]
    if s != 0:
        count = count + 1

    print('第%d行主题和回复的相似度为：%.2f' % (i, s))

print('回复相关性:', count / nrows)
```

3.3.2 LDA 获取回复主题

```
import re

import jieba
from openpyxl import load_workbook
import gensim

if __name__ == "__main__":

    # 读入停用词
    stop_words = []

    f = open("../清洗规则/stopword.txt", 'r', encoding="gbk")

    sourceInLine = f.readlines()
    for line in sourceInLine:
        temp = line.strip('\n')
        stop_words.append(temp)
    f.close()

    # 读取文本

    wb = load_workbook("../清洗后数据及输出数据/附件4_第三问_清洗后.xlsx")

    sheet = wb.get_sheet_by_name("Sheet1")
    sentences = []  # list 中的每个元素也是一个 list,为每句话的分词结果
    for line in sheet['B']:
        try:
            segs = jieba.lcut(line.value)
            segs = filter(lambda x: len(x) > 1, segs)
            segs = filter(lambda x: x not in stop_words, segs)
            words_line = []
            # 将分词后句子中的纯数字去掉
            for word in segs:
                if word.isdigit():
                    continue
                else:
                    words_line.append(word)
            sentences.append(words_line)
        except Exception:
            print(line)
        continue

    # 词袋模型
    dictionary = gensim.corpora.Dictionary(sentences)
    corpus = [dictionary.doc2bow(sentence) for sentence in sentences]
    # 训练四个主题出来
    lda = gensim.models.ldamodel.LdaModel(corpus=corpus,
id2word=dictionary, num_topics=5)
    first = lda.print_topic(0, topn=10)
    print(first)  # 输出第一个topic 出来, 用十个词表示
    a = re.findall('[\u4e00-\u9fa5]+', first)  # 只要字符串中的中文
    a = "".join(a)
    print(a)
    # 把所有主题都输出出来

    t = open('../清洗后数据及输出数据/附件4_第三问_主题提取.txt', 'w')

    t.write(str(a))
    t.close()
```

3.3.3 TF-IDF 完整性评价

```
import warnings

import jieba
from gensim.corpora import Dictionary
from gensim.models import TfidfModel
from gensim.similarities import SparseMatrixSimilarity

warnings.filterwarnings('ignore')  # 取消警告

import xlrd


data = xlrd.open_workbook('../原始数据/附件4.xlsx')

st = data.sheets()[0]
nrows = st.nrows
```

```
st2 = st.col_slice(5, start_rowx=0, end_rowx=None)

with open("../清洗后数据及输出数据/附件4_第三问_主题提取.txt", "r") as f:

    data = f.readline()
a = data
count = 0.0
sumAvg = 0.0
for i in range(1, nrows):
    b = st2[i].value
    texts = [b, '']
    keyword = a
    # 1、将【文本集】生成【分词列表】
    texts = [jieba.lcut(text) for text in texts]
    # 2、基于文本集建立【词典】，并获得词典特征数
    dictionary = Dictionary(texts)
    num_features = len(dictionary.token2id)
    # 3.1、基于词典，将【分词列表集】转换成【稀疏向量集】，称作【语料库】
    corpus = [dictionary.doc2bow(text) for text in texts]
    # 3.2、同理，用【词典】把【搜索词】也转换为【稀疏向量】
    kw_vector = dictionary.doc2bow(jieba.lcut(keyword))
    # 4、创建【TF-IDF 模型】，传入【语料库】来训练
    tfidf = TfidfModel(corpus)

    # 5、用训练好的【TF-IDF 模型】处理【被检索文本】和【搜索词】
    tf_texts = tfidf[corpus]  # 此处将【语料库】用作【被检索文本】
    tf_kw = tfidf[kw_vector]
    # 6、相似度计算
    sparse_matrix = SparseMatrixSimilarity(tf_texts, num_features)
    similarities = sparse_matrix.get_similarities(tf_kw)
    s = similarities[0]

    print('第%d行回复格式完整性为：%.2f' % (i, s))

    sumAvg = sumAvg + s
    if s > (sumAvg / i):
        count = count + 1
    print(tf_kw)


print('回复完整性:', count / nrows)
```

4 模型的实用性

众所周知，政府的信息公开和信息响应速度一直是许多通过移动平台或者广域网络进行即时通讯和动态发布的网络用户所诟病，这集中反应在微型博客的动态留言和转发的评论当中，而政府平台的留言处理的不及时和回复的不准确是导致这些情况出现的关键部分，因此政府部门及时的回复相关网络用户和广大公民的关切，第一时间了解当前的热点问题并作出回应是十分有必要的，在本次 C 题中我队训练出的分类模型通过交叉验证和 F 检验得出的平均准确率均在 90%左右，响应时间在模型训练完成后<1 秒，对政府处理群众留言或者新型社交媒体平台上的相关评论及转发进行及时的分类处理具有非常重要的意义。

其次，在热点问题分析上，本队所训练的模型可以对大量的留言和评论的主题进行提取，并可以较为准确的提取聚类，将相似的留言或者评论聚合，提取共同点，按照其点赞或者反对进行加权排序，并按照时间序列进行分类汇总，可以方便政府对当前的热点问题进行剖析，这对政府的及时热点应急处理响应能力会起到非常大的提示。

政府对社会问题的回复常常成为普通民众关注的焦点，很多不合时宜或者不小心错序的回复常常引发大量网络使用者和普通民众讨论的热点话题，从而降低政府的社会公信力，造成极其不好的负面影响，而利用模型在对外公布前进行回复评价，并根据评价结果对回复进行修正或发布后，可以在一定程度上规避因偶然的意外照成的回复错序等情况，是有十分重要的实用性。

5 参考文献

[1]. 余传明. 基于深度学习的词汇表示模型对比研究[DOI].2019.12.22. 10.11925/infotech.2096-3467

[2]. 李丽. 基于 CHARLS 数据的中老年人口腔卫生服务利用现状及影响因素分析. 中国公共卫生.Chinese Journal of Public Health.ISSN：1001-0580.中文核心期刊. 2019-12-24

[3]. 吴琼/余文铖/洪海生/喻蕾/段炼/尚明远/刘哲. 基于 XGBoost 算法的配网台区低压跳闸概率预测. 中国电力 Electric Power.2020 年 04 期.ISSN：1004-9649. 中文核心期刊

[4]. 张明生/邓少灵. 基于 MBUT-LDA 主题模型的微博文本挖掘研究. 电子商务 2019,(07),70-71

[5]. 王涛/李明. 基于 LDA 模型与语义网络对评论文本挖掘研究.重庆工商大学学报（自然科学版）2019,36(04),9-16 DOI:10.16055/j.issn.1672-058X.2019.0004.002

[6]. 单国栋/肖彦翠/王皓. 基于主题模型的中外期刊文献挖掘对比研究. 长春大学学报. Journal of Changchun University.2019 年 06 期. ISSN：1009-3907

[7]. 基基伟. Python+gensim-文本相似度分析.CSDN.YellowPython. 2018-07-12