“智慧政务”中的文本挖掘应用

摘　要

随着时代的变化，政府收集群众的意见方式也在改变，从以前使用填写纸张的方式到现在使用网络平台收集意见。文本数据量的增大，给整理数据的工作人员带来了难度。本文使用python语言处理自然文本，对数据做分词、去停用词和统计等操作，计算词频和逆文本频率，把留言详情做中文分词并提取特征词或找出相同事件的关键词，使用TF − IDF模型对文本做相似性计算，定义热点指数，且做出评价结果。答复意见从相关性、完整性、可解释性的角度，来评判答复意见的质量。

针对问题一，首先对附件1中各标签，做提取去重的处理，更直观的看到一级标签、二级标签、三级标签下的小标签。其次对附件2中的留言详情用python语言做中文分词并统计词频，把无意义的词语做为”停用表”。编写python语言代码，把的留言详情做中方分词并提取特征词把最后的结果存放在附件2_1.xls中，利用TD − IDF模型对附件2_1.xls做文本相似性计算，不用类别用不同的标号表示，判断标号是否一致，标号一致的的统为一类　结果保存再附件2_2中，其次使用F − score对分类方法进行评价。

针对问题二，首先将附件3中的留言主题和留言详情合成为留言内容，做数据预处理操作，文本分词，去除停用词。其次计算TF和IDF，将一条举报记录做为词频，全部举报记录为逆文本频率，使用贪婪算法匹配词语，尽可能的匹配最大长度。然后找出相同事件的关键词。最后使用热点问题留言明细表中的举报条数、点赞数与反对数来定义热点指数，得出了热点最高的前5项，分别是丽发新城、伊景园滨河苑、58车贷、魅力之城、经济学院。

针对问题三，使用 python 语言将附件 4 中的答复意见列提取到文本文件中。后使用 RUBER 评价方法，依据答复的完整性、可解释性等角度，来评价回复的质量，得出回复的质量比较好。

关键字：文本挖掘　自然语言　　TF − IDF算法　RUBER评价方法

一、问题重述

1.1 背景知识

近年来，各种问政平台的出现，更加方便政府了解各类社情民意，一般在微博、微信、阳光热线等网络问政平台上了解民意。同时也带来了一系列问题，随着文本数据量的增大，给以往整理数据的人员带来了极大的挑战。但随着大数据、人工智能、云计算等技术发展得越来越成熟，也提供了一些便利，建立基于自然语言处理技术的智慧政务，有利于大量数据的处理，加快处理的速度，也可以提升政府的管理水平和施政效率，更是一种治理方法的新趋势。因此，政府对这方面有一定的重视性。

1.2 要解决的问题

问题一：将群众留言分类，分别划分为一级标签、二级标签、三级标签。根据附件2的数据，建立一级标签分类模型，以留言内容为分类依据。使用评价方法对分类方法进行评价。

问题二：根据附件3，将留言信息进行分类，且定义合理的热度评价指标，得出评价结果。按照题目示例数据，将数据整理成表格形式，分别为“热点问题表.xls”、“热点问题留言明细表.xls”。

问题三：根据附件4中的“答复意见”，从答复的各个角度对答复意见的质量给出评价方案，且尝试去实现它。

二、问题分析

首先将留言进行分类，分别分为一级标签、二级标签、三级标签。根据附件2的数据建立关于留言内容的一级标签分类模型，并使用方法进行评价。其次，根据附件3对留言内容进行归类，定义合理的评价指标，得出评价的结果。最后，针对附件4中的答复意见，从答复的相关性、完整性等方面对答复意见的质量给出一套评价方案。

2.1 问题一的分析

对留言信息分类，分类参考附件1，划分为一级标签、二级标签、三级标签，对每个标签下的小标签归类，去掉重复的，将唯一的文本呈现出来，保存为附件。去停用词做分词，统计出现最多的词语，使用$TF-IDF$算法，对留言内容的一级

标签分类建立模型，且对分类方法进行评价。

2.2 问题二的分析

根据附件3进行操作，对“留言主题”和“留言详情”做合成操作，找出出现次数最多的词频。再做分词操作，得出分词文档和统计文档。其次，筛选出前5项，使用正则表达式之贪婪与非贪婪算法，定义一个合理的热点评价指标，从而得出热点问题留言明细的前五项，将之保存为文件。

2.3 问题三的分析

从附件4中“答复意见”这一列入手，从相关性、完整性、可解释性等方面，对答复意见的质量给出一套评价方案。

三、模型假设

1. 假设所给数据准确，没有逻辑问题。
2. 假设答复的完整性与回复长度成正比。
3. 假设答复意见有可解释性、相关性、完整性。

四、符号说明

符号	意义
$\mathrm{TF}-\mathrm{IDF}$	词频-逆向文件频率
TF	词频
IDF	逆向文件频率
d_j	
$n_{i,j}$	文件中出现的总数
$\sum_k n_{k,\ j}$	所以词汇中出现的次数总和
|D|	语料库中的文件总数
$\|\{\mathrm{j}: t_i \in d_j\}\|$	包含t_i的文件数目
S	热点评价指标
t	同事件举报数量
d	点赞数
f	反对数
Precision	精确率

Recall	召回率
TP	相关数
FP	不相关数

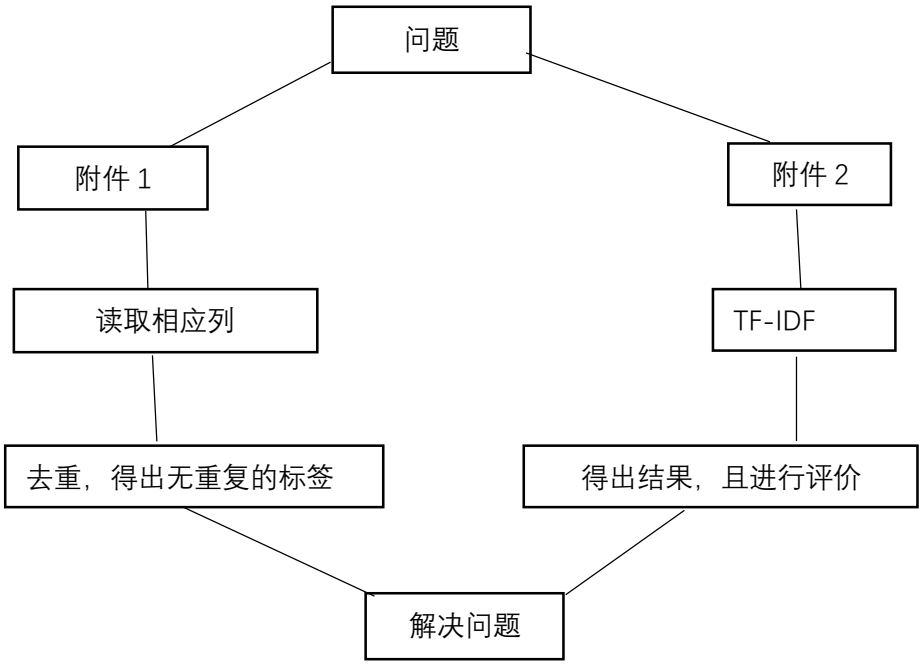
五、模型的建立与求解

5.1 问题一

5.1.1 思路分析

首先对附件1进行处理，将附件1中的一级标签、二级标签、三级标签下的小标签做提取，然后对小标签做去重处理。其次，为附件2中的留言内容做一级标签分类模型，使用$TF-IDF$[1]模型，进行处理。最后对分类方法进行评价，使用$TF-IDF$的相似性评价。

其思维导图如图一（图一　问题一思维导图）所示：



图一　问题一思维导图

5.1.2 解决问题

```
f= open('一级标签直接读取.txt', 'w',encoding='utf-8')
inpath = '附件1.xlsx'  # excel文件所在路径
extract(inpath)
f.close()

# 将读取出来的数据变成列表形式，并去重
f = open('一级标签直接读取.txt', 'r',encoding='utf-8')
a=[i for i in f]
lst2 = list(set(a))
print(lst2)
```

（一）附件1的处理

（1）使用python语言，将附件1中的“一级标签”、“二级标签”、“三级标签”列分别从excel中取出来，用变量存放。

（2）使用循环遍历数据，直到循环到末尾，为空便停止循环。将数据逐个存放到文件中。

（3）将读取出来的数据，变成列表形式。对列表进行去重操作，示例代码如下图（图二 标签处理图），具体代码见附录1。

图二　标签处理图

（4）分别对“一级标签”、“二级标签”、“三级标签”进行去除重复之后得出下表（表1　无重复标签表）：

表1　　无重复标签表

一级标签	二级标签	三级标签
国土资源	社会治安	体育事业
政法	民间组织	疫情防控
…	…	…
经济管理	企业破产	运输购销
农村农业	社保基金管理	文化艺术

从去除重复数据的表中，可以很直观的看到“”一级标签”、“二级标签”、“三级标签”下的信息（具体内容见附录2）。

（二）模型的建立与求解

本题使用TF—IDF算法模型来解决问题，TF—IDF是一种信息检索与文本挖掘的加权技术。TF—IDF用以评估字词或文件集或语料库在某一份文件中的重要性。

如果某个单词在一篇文章中出现的次数高，但在其他文章中很少出现，则认为具有很高的区分程度，适合用于分类。关于TF—IDF算法的具体使用步骤如下所示。

（1）TF表示关键字在文本中出现的频率，TF得出的数字一般会进行归一化，公式如下：

$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

即

$$TF_W = \frac{在某一类中词条W出现的次数}{该类中所有的词条数目} \quad (1)$$

(2)IDF:某一特定词语的IDF,可以由总文件数目除以包含该词语的文件的数目,再将得到的商取对数得到.若包含的词条t的越少, IDF越大,则说明词条具有很好的类别区分能力。公式如下:

$$idf_i = log\frac{|D|}{|\{j: t_i \in d_j\}|}$$

即

$$\mathrm{IDF} = \log\left(\frac{语料文档总数}{包含词条w的文档数+1}\right) \quad (2)$$

(3)TF * IDF:某一文件内的高词语频率,以及该词语在整个集合的低文件频率,可以产生出高权重的TF—IDF。计算公式如下:

$$\mathrm{TF-IDF} = \mathrm{TF} * \mathrm{IDF} \quad (3)$$

用TF—IDF算法解决问题的步骤如下:

（1）首先将附件 2 中的“留言详情”进行中文词频统计。使用 python 语言

将“留言详情”列读取出来，放入文本文件中，以便于后续操作。具体代码见附录 3。

（2）对提取出来的“留言信息”做分词处理，去掉停用词，并且将某个词语出现的次数进行统计，示例代码如下图（图三　词频统计图），具体代码见附录 4。

```
import xlrd
import jieba.analyse
from xlutils.copy import copy
wordListTip10=[]
stop = []
file_stop =r'停用词.txt'  #  打开txt文件
with open(file_stop,'r',encoding='utf-8-sig') as f:
    lines = f.readlines()
    for line in lines:
        lline = line.strip()# 去除\n
        stop.append(lline)
fp = xlrd.open_workbook('附件2.xlsx', 'r')
sheel= fp.sheet_by_name('Sheet1')
new_excel = copy(fp)
ws= new_excel.get_sheet(0)
c=sheel.nrows
```

图三 词频统计图

将程序运行之后得出下图（图四 结果图），将结果存储为附件2_1.xls（在附件1中）。

图四　结果图

（3）把附件2_1.xls用TD－IDF做文本相似性计算一下，把每一个类别比作不同的标号如下表（表2 文本相似级别表）所示，用代码判断如果相似度标号一样，则把相似度标号一样的统为一类 结果保存再附件2_2.xls中。

表2　　文本相似级别表

一级标签	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
相似度编码	1	2	3	4	5	6	7

具体操作代码见图五（图五　具体操作图），具体代码见附录5。

```
while counter:
    if(counter == 1 | counter == 2011 | counter == 2949 | counter == 3561 | counter == 5151 |
    counter == 7120 | counter == 8334 | counter == 9211):
        data1 = shee1.col(4)[counter].value
        data1 = data1.strip()
        seg_list = jieba.cut(data1, cut_all=False)
        wordListTip10.append(seg_list)
    counter = counter + 1
    if (counter == c):
        break
dictionary = corpora.Dictionary([wordListTip10])
# print(dictionary.token2id)
corpus = [dictionary.doc2bow([doc]) for doc in wordListTip10]
# print(corpus)
while counter1:
    data2 = shee1.col(4)[counter1].value
    data2 = data2.strip()
    seg_list = jieba.cut(data2, cut_all=False)
    tfidf = models.TfidfModel(corpus)
    # print(tfidf)
    index = similarities.SparseMatrixSimilarity(tfidf[corpus], num_features=len(dictionary.keys
    sim = index[tfidf[seg_list]]
    sim_sorted = sorted(enumerate(sim, 1),key=lambda x: -x[1])
    print(sim_sorted)
    ws.write(counter1, 5, sim_sorted)
    counter1=counter1+1
    if ( counter==c ):
        break
```

图五　具体操作图

具体结果如下图所示（图六　结果图），具体结果见附录6：

留言编号	留言用户	留言主题	留言时间	留言详情	一级标签	相似度编码
24	A0007401	A市西湖建	2020/1/6 1	路段 A3 未管 文明城市 车流 安全隐患 西行 加油	城乡建设	1
37	U0008473	A市在水一	2020/1/4 1	护栏 在水一方 电等 过往行人 锈迹斑斑 烂尾 四楼	城乡建设	1
83	A0006399	投诉A市A	2019/12/30	车位 停车 停车费 A1 公摊 程明 小区业主 社区 征	城乡建设	1
303	U0007137	A1区蔡锷	2019/12/6	不洗 A1 A2 区华庭 水是 健康 霉味 致癌物 水箱	城乡建设	1
319	U0007137	A1区A2区	2019/12/5	不洗 A1 A2 区华庭 水是 健康 霉味 致癌物 水箱	城乡建设	1
379	A0001677	投诉A市盛	2019/11/28	物业公司 耀凯 水电费 业委会 物业费 为所欲为 问	城乡建设	1
382	U0005806	咨询A市楼	2019/11/27	供暖 近年 楚江 气候 具体位置 片区 阴冷 楼盘 常	城乡建设	1
445	A0001920	A3区桐梓	2019/11/19	停水 自来水厂 胡书记 A3 小区业主 桐梓 业委会	城乡建设	1
476	U0003167	反映C4市	2019/11/15	垃圾处理 物业公司 代收 垃圾 环卫局 确实 您提	城乡建设	1
530	U0008488	A3区魏家	2019/11/10	绿化带 A3 人管 社区 商业街 破烂不堪 停车 每次	城乡建设	1
532	U0008488	A市魏家坡	2019/11/10	绿化带 A3 人管 社区 商业街 破烂不堪 停车 每次	城乡建设	1
673	A0008064	A2区泰华-	2019/10/24	居委会 业委会 非法 滨江 小区业主 泰华 社区 一	城乡建设	1
994	U0005196	A3区梅溪	2019/9/18	自来水 溪湖 停水 用水 恳请 壹号 御湾 就会水 转	城乡建设	1
1005	U0006509	A4区鸿涛	2019/9/18	金晖 交房 限电 鸿涛 延迟 停水 第二次 同意 协议	城乡建设	1
1110	A0009977	地铁5号线	2019/9/9 1	停电 每次 号线 丽路 锦楚 星城 来次 断电 被困 三	城乡建设	1
1309	U0005083	A6区润和	2019/8/21	高压线塔 用电 A6 周边 保障 说好 电力局 区润 改	城乡建设	1
1440	A0003288	A市锦楚国	2019/8/6 1	停电 线路 A5 朝晖路 锦楚 三区 国家电网 新城 炎	城乡建设	1

图六　结果图

5.1.3 评价方法

对于以上所得结果，本文采用$F-score$[2]评价方法。

在理想的情况下，做到精确率和召回率指标都高，这是最好的，其计算公式为：

$$F-Sxore=(1+\beta^2)\frac{Precision*Recall}{(\beta^2*Precision)+Recall}* \quad (4)$$

精确率的计算公式：

$$Precision=\frac{TP}{TP+FP} \quad (5)$$

召回率的计算公式：

$$Recall=\frac{TP}{TP+FN} \quad (6)$$

$F-score$中β值是用来起到两个指标的平衡作用，在$F-score$计算的权重中，取值有三种情况。

1） 取 1, 表示Precision与Recall一样重要。

2） 取小于 1, 表示Precision比Recall重要

3） 取大于 1, 表示Recall比Precision重要

一般情况下， β 取 1，则此时 F-score 的计算公式为：

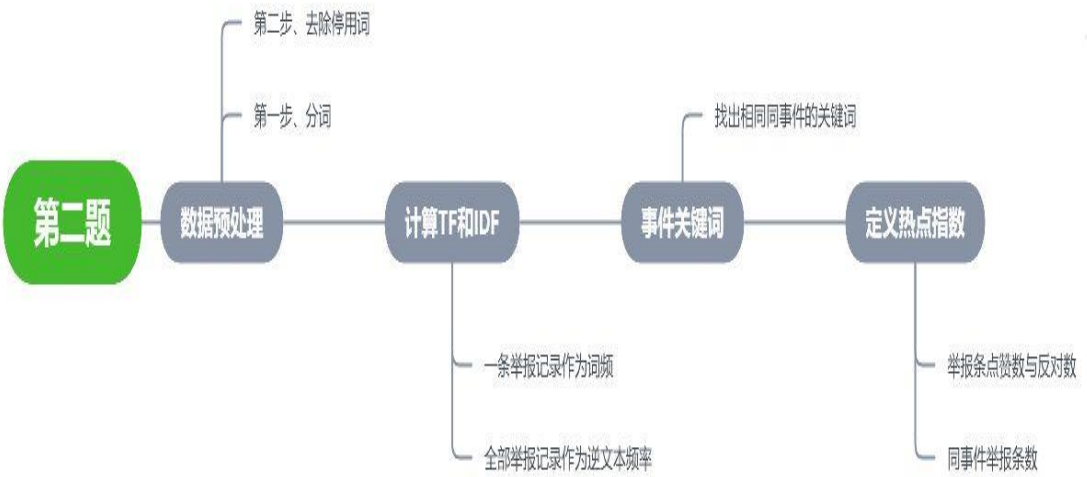
$$F-Sxore=(1+\beta^2)\frac{Precision*Recall}{(\beta^2*Precision)+Recall}* \quad (7)$$

通过数据显示，本文所采用分类的方法是准确的，并且存在一定的准确率。能够对数据进行合理的分类。

5.2 问题二

5.2.1 思维分析

首先对数据做预处理操作，将留言主题和留言详情合成为一个文档，则为留言内容，对留言内容做中文分词并且去掉停用词。其次计算TF 和 IDF，一条记录为词频，全部举报记录为逆文本频率，然后找出相同事件的关键词，最后定义热点指数，评价结果。其思维导图如下（图七　第二题思维导图）：



图七　第二题思维导图

5.2.2 模型的建立与求解

问题二采用的是TF—IDF算法，IF－IDF算法的使用方法在5.1.2　模型的建立与求解中，对应第二题的具体操作如下：

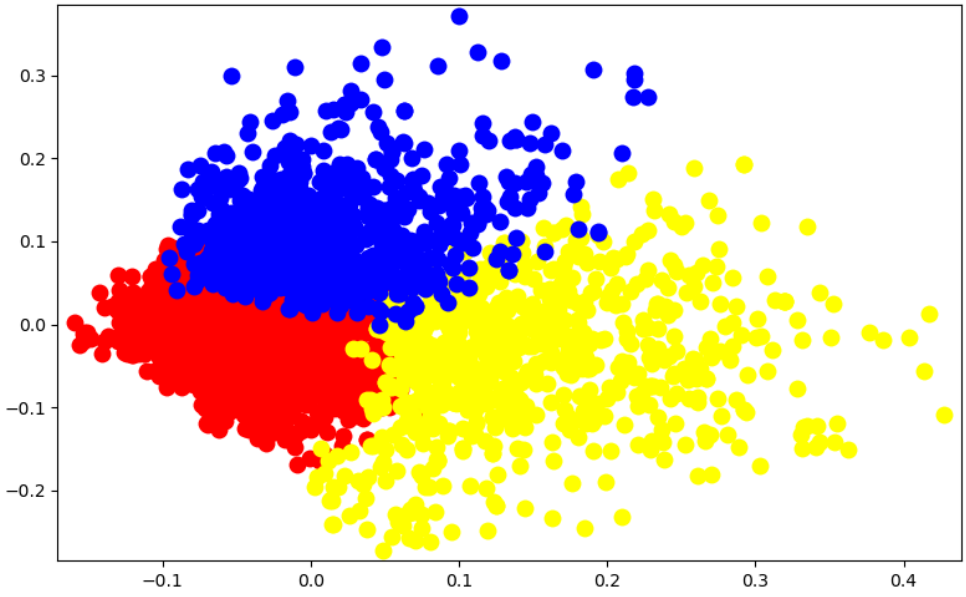
（1）首先对数据做预处理操作，将附件3中的留言主题和留言详情合成为留言内容，然后做中文分词的操作。直接分词会发现一些标点符号出现的次数比较多。在这种情况下，我们就应该使用停用词，将无关的信息去除，不做统计，用python语言的操作如下图（图八　第二题示例代码），具体操作代码见附录7。

```
# coding:utf-8

import ...

# 清洗文本
def clearTxt(line:str):
    if(line != ''):
        line = line.strip()
        # 去除文本中的英文和数字
        line = re.sub("[a-zA-Z0-9]", "", line)
        # 去除文本中的中文符号和英文符号
        line = re.sub("[\s+\.\!\/_,$%^*(+\"\';:.“”.]+|[+——！，。？?、~@#￥%……&*（）]+", "", line)
        return line
    return None

#文本切割
def sent2word(line):
clearTxt() › if (line != '')
```

图八　第二题示例代码

程序运行完毕之后，会出现一个cut.txt 文档（文档在附件3中），该文档是分词文档，已经过处理的。

（2）将分词得出的四千多条记录做为IDF的语料库，以便用来进行下一步操作，通过留言详情和留言主题的 TF 判断投诉，并且找出语料库中最大的数据，存储，将得出的前五项绘制散点图（图九　数据散点图），具体实现代码见附录8。



图九 数据散点图

（3）为去除标点符号及其与热点问题无关之类无用的数据，应进行处理。

其中所采用的停用词为“去年、安全、食品、一个、中国、自己……”等词，停用词文档为stop.txt,具体内容在附件3中。其代码如下：

```
import xlrd
import jieba

data = xlrd.open_workbook('附件 3.xlsx')
stop = open('stop.txt', 'r', encoding='utf-8-sig')
stop = stop.read().split()
table = data.sheet_by_name('Sheet1')
UrlList = table.col_values(2)
cipin = {}
shu = 0
print(len(UrlList))
#.......
```

具体代码件附录9。

将统计所得出的词语和出现的次数存入csv文件中，具体内容在附件3中cipin3.csv文件中。

（4）通过用程序筛选，将出现次数最高的前五项，做为匹配的条件，使用正则表达式之贪婪[3]与非贪婪模式。

贪婪匹配：正则表达式一般趋向于最大长度匹配。

非贪婪匹配：匹配到结果就行，少量的匹配字符。

在问题二的应用中，用到的是贪婪匹配，最大长度的匹配，有利于我们处理数据，更容易找到热点问题。

（5）经过上面的步骤之后，将留言信息按照某一时间段的特定地点或人群归类。对于热度评价指标，我们的定义方法按照已经得出的“热点问题留言明细表”中的点赞数、反对和同事件举报数来进行计算。具体计算公式如下：

$$S = \mathrm{t} + \frac{d+f}{100} \qquad (t > 5) \tag{8}$$

热点即广大群众最关注的新闻或者信息，某一时间段对最引人注目的某个地方或问题。所以热度评价指标的高低也是是看一个事件引入注目的程度。

时间范围则是某一事件最开始的时刻到人们结束讨论的时刻。这里我们将与生活中的常识结合，去时间轴的起点和终点做为事件范围。

地点/人群：统计出现在时间段内出现最多的地点/人群。

问题描述使用贪婪算法，取得匹配的最长度。

因此，热点中的前五项为（表 3　热点问题表）：

表3 热点问题表

热度排名	问题ID	热度指南	时间范围	地点/人群	问题描述
1	1	55.51	2019/11/2 至 2020/1/26	丽发新城	小区附近工厂扰民污染环境
2	2	54.54	2019/7/28 至 2019/9/1	伊景园滨河苑	该开发商捆绑销售停车位
3	3	37.6	2019/1/11 至 2019/7/8	58 车贷	58 车贷诈骗调查不满
4	4	6.6	2019/7/21 至 2019/9/25	魅力之城	小区临街餐饮店油烟噪音扰民
5	5	5.4	2019/04/28 至 2019/11/22	经济学院	经济学院强制定点实习

通过热点问题表，我们可以很直观的看出各问题ID的热度指南，还有该问题出现的时间段。为了更加清楚、直观的看到问题明细，因此，请看下表（表4 热点问题留言明细表）。

表 4　　热点问题留言明细表

问题ID	留言编号	留言用户	留言主题	留言详情	点赞数	反对数
2	188801	A909180	投诉滨河苑	尊敬的张市	0	0
1	188809	A909139	A市万家丽	A市万家丽	1	0
1	189950	A909204	投诉A2区丽	我是A2区丽	0	0
1	190108	A909240	丽发新城小	丽发新城小	1	0
2	190337	A0009051	关于伊景园	投诉伊景园	0	0
1	190523	A0007284	A市丽发新	领导您好！	0	0
1	190802	A0007263	A市丽发小	发同投资有	0	0
2	191001	A909171	A市伊景园	商品房伊景	12	1
2	192739	A909188	请政府救救	实在搞不懂	0	0
3	194343	A0001061	承办A市58	胡书记：您	733	0
2	195511	A909237	车位捆绑违	对于伊景园	0	0
2	195995	A909199	关于广铁集	尊敬的市政	0	0
2	196264	A0009508	投诉A市伊	A市伊景园	0	0
1	199379	A0009224	A2区丽发新	A市A2区丽	0	0
2	200085	A0001042	A市市政建	我是广铁的	9	2
1	203393	A0005306	A市丽发新	发同投资有	2	0
2	204960	A909192	家里本来就	我是广铁集	0	0
2	205277	A909234	伊景园滨河	广铁集团强	1	0
2	205982	A909168	坚决反对伊	我坚决反对	2	0
2	206355	A909189	广铁集团售	一、做为A	0	0
2	207243	A909175	伊景园滨河	您好！A市	0	0
1	208285	A909205	投诉小区附	尊敬的领导	24	0
1	208714	A0004201	A2区丽发新	尊敬的领导	4	0
2	209506	A909179	A市武广新	您好！由A	0	0
2	209571	A909200	伊景园滨河	广铁集团铁	0	0
2	212323	A0002070	广铁集团要	尊敬的领导	0	0

Sheet1

因文字过多的缘故，显示上有很大的问题，使用图片来观看。具体的“热点问题表”、“热点问题留言明细表”分别对应作品附件中的“热点问题表.xls”、“热点问题留言明细表.xls”。

从热点问题留言明细表中我们可以明确的看出留言的用户、留言的主题与详情，同时也可以看到该条留言的点赞数和反对数，点赞即代表同意观点，反对即代表不同意观点。

5.3 问题三

5.3.1 思路分析

将附件 4 中的 “答复意见”列，使用 python 语言提取出来存入文本文件中，后使用 RUBER 评价方法，从答复意见的可解释性、完整性等角度来评价答复的质量。

5.3.2 解决问题

（一）使用python语言将答复意见列提取出来，并存放到“答复意见.txt”。具体操作代码如下（源文件存放于附件 4 中）：

```
import xlrd


def extract(inpath):
    data = xlrd.open_workbook(inpath, encoding_override='utf-8')
    table = data.sheets()[0]  # 选定表
    nrows = table.nrows  # 获取行号
    ncols = table.ncols  # 获取列号

    for i in range(1, nrows):  # 第 0 行为表头
        alldata = table.row_values(i)  # 循环输出 excel 表中每一行，即所有数据
        result = alldata[5]  # 取出表中第二列数据
        f.write('"'+result+'"'+'\n')
        print(result)

f= open('答复意见.txt', 'w',encoding='utf-8')
inpath = '附件 4.xlsx'  # excel 文件所在路径
extract(inpath)
f.close()
```

（2）使用RUBER[4]来评价回复的质量。RUBER是一种无监督对话系统回复质量的评价方法。

首先要获取句子的向量表示，其次计算余弦，计算公式如下：

$$S\ \ R(r.\hat{r}) = \cos(U_r.\widehat{U}_{\hat{r}}) = \frac{U_r \tau \widehat{U}_{\hat{r}}}{\|U_r\| \cdot \|\widehat{U}_{\hat{r}}\|} \tag{9}$$

将SR和SU度量值做归一化处理，再将归一化的特征进行结合。其做归一化的公式如下：

$$\hat{S} = \frac{S - \min(\acute{S})}{\max(\acute{S}) - \min(\acute{S})} \tag{10}$$

最后将归一化的度量值，取平均、取最大/小值，算术平均带入问题三计算。

六、模型的评价与推广

6.1 模型的评价

6.1.1 模型的优点

（1）使用 python 语言能更好的处理自然文本语言。

（2）TF—IDF 算法使用较为简单，并且很容易实现。能处理相关性系数，从而更容易解题。

（3）贪婪算法能尽最长度的匹配，匹配信息面广。

6.1.2 模型的缺点

TF—IDF 没有考虑词语的语义信息，无法处理一词多意的现象。计算上可能出现一些小误差。

6.2 模型的改进与推广

应当多使用，更容易发现问题，避免产生问题的地方，尽可能的解决问题。

七、参考文献

[1]夏桂梅,赵晋彬.一种基于混沌搜索和贪婪模式的混合优化算法[J].太原科技大学学报,2020,41(02):154-157+164.

[2]张波,黄晓芳.基于 TF-IDF 的卷积神经网络新闻文本分类优化[J].西南科技大学学报,2020,35(01):64-69.

[3]谢娟英,郑清泉,吉新媛.F-score 结合核极限学习机的集成特征选择算法[J].陕西师范大学学报(自然科学版),2020,48(02):1-8.

[4]李魁晓,顾继东.红树林细菌 Rhodococcus ruber1K 降解邻苯二甲酸二丁酯的研究[J].应用生态学报,2005(08):1566-1568.

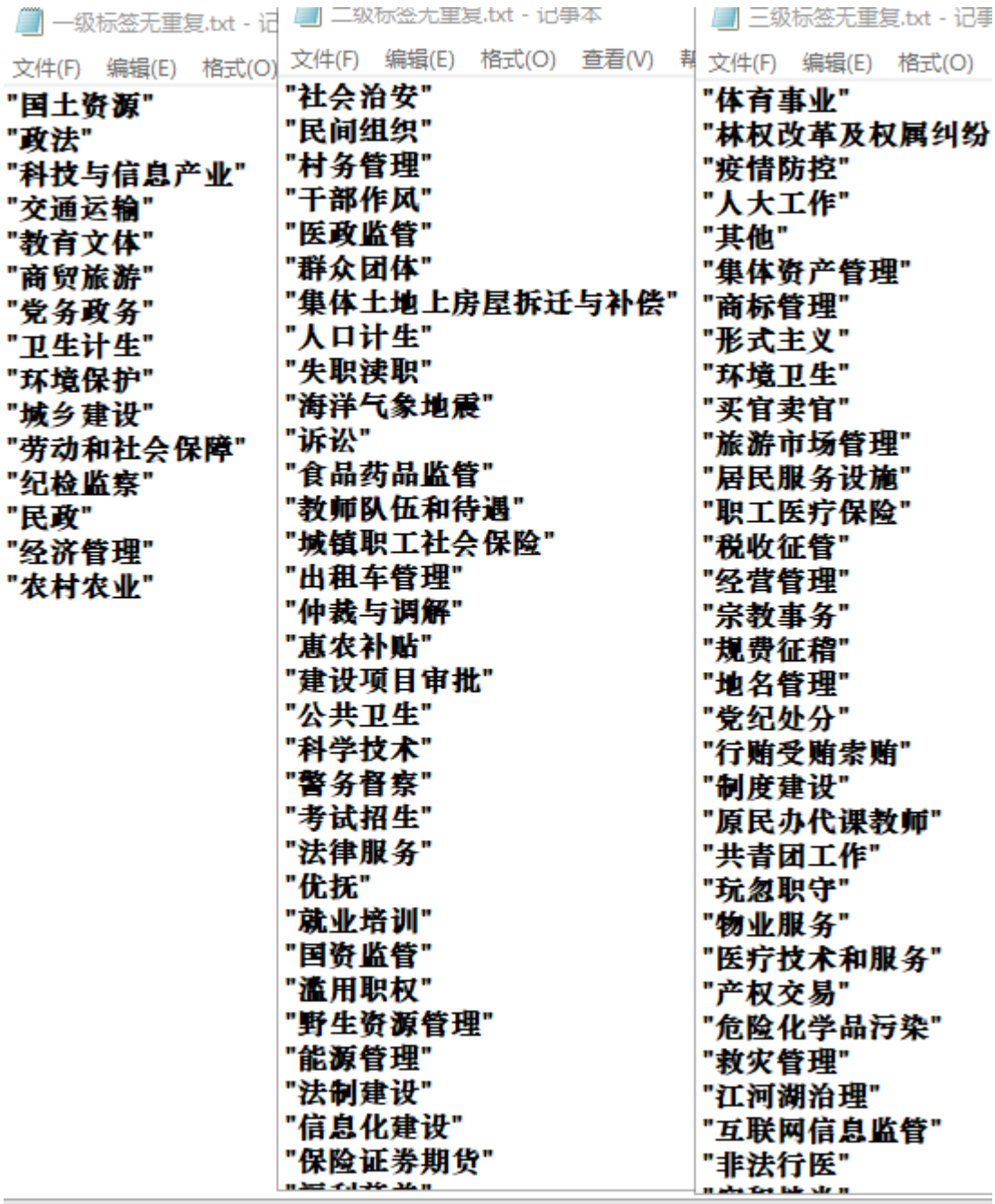
附录

附录 1：

```
import xlrd


def extract(inpath):

    data = xlrd.open_workbook(inpath, encoding_override='utf-8')
    table = data.sheets()[0]  # 选定表
    nrows = table.nrows  # 获取行号
    ncols = table.ncols  # 获取列号

    for i in range(1, nrows):  # 第 0 行为表头
        alldata = table.row_values(i)  # 循环输出 excel 表中每一行，即
所有数据
        result = alldata[0]  # 取出表中第二列数据
        f.write('"'+result+'"'+'\n')

f= open('一级标签直接读取.txt', 'w',encoding='utf-8')
inpath = '附件 1.xlsx'  # excel 文件所在路径
extract(inpath)
f.close()

# 将读取出来的数据变成列表形式，并去重
f = open('一级标签直接读取.txt', 'r',encoding='utf-8')
a=[i for i in f]
lst2 = list(set(a))
print(lst2)
f.close()

# 写入无重复文本文件
f1 = open('一级标签无重复.txt', 'w',encoding='utf-8')
for i in lst2:
    print(i)
    f1.write(i)
```

具体代码和文件见附件 1。

附件 2：



具体文件见附件 2。

附录 3：

```
import xlrd

def extract(inpath):
```

```
    data = xlrd.open_workbook(inpath, encoding_override='utf-8')
    table = data.sheets()[0]  # 选定表
    nrows = table.nrows  # 获取行号
    ncols = table.ncols  # 获取列号

    for i in range(1, nrows):  # 第 0 行为表头
        alldata = table.row_values(i)  # 循环输出 excel 表中每一行，即
所有数据
        result = alldata[4]  # 取出表中第二列数据
        f.write(result)
        print(result)

f= open('留言信息.txt', 'w',encoding='utf-8')
inpath = '附件 2.xlsx'  # excel 文件所在路径
extract(inpath)
f.close()
```

附录 4：

```
import jieba
import gensim
txt = open("留言信息.txt", encoding="utf-8").read()
#加载停用词表
stopwords = [line.strip() for line in open("CS.txt",encoding="utf-
8").readlines()]
words  = jieba.lcut(txt)
counts = {}
for word in words:
    #不在停用词表中
    if word not in stopwords:
        #不统计字数为一的词
        if len(word) == 1:
            continue
        else:
            counts[word] = counts.get(word,0) + 1
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(20):
    word, count = items[i]
    print ("{:<10}{:>7}".format(word, count))
```

结果：

附录 5：

```
import  xlrd
import  jieba.analyse
from xlutils.copy import copy
wordListTip10=[]
stop = []
file_stop =r'停用词.txt' #  打开 txt 文件
with open(file_stop,'r',encoding='utf-8-sig') as f:
    lines = f.readlines()
    for line in lines:
        lline = line.strip() # 去除\n
        stop.append(lline)
```

```
fp = xlrd.open_workbook('附件 2.xlsx', 'r')
sheet1= fp.sheet_by_name('Sheet1')
new_excel = copy(fp)
ws= new_excel.get_sheet(0)
c=sheet1.nrows
counter=1
while counter:
    outstr= ""
    data1=sheet1.col(4)[counter].value        # 读取 counter 行，评论列单
元格内容
    seg_list = jieba.cut(data1, cut_all=False)      # 对读取的 text 评论
分词
    for word in  seg_list:    #去掉停用词
        if word not in stop:
            if word !='\t':
                outstr += word
                outstr +=' '
    data2 = sheet1.col(4)[counter]=str("
".join(jieba.analyse.extract_tags(outstr,topK=10,withWeight=False)))
# 把分词结果赋给相应的单元格
    ws.write(counter, 4, data2)
    counter=counter+1
    if (counter==c):
        break
new_excel.save('附件 2_1.xls')
```

具体代码和结果见附件 2

附录 6：

具体结果见附件 2_2.xls

附录 7：

```
# coding:utf-8

import xlrd
import jieba
import pandas as pd
import codecs
import string
import re

# 清洗文本
def clearTxt(line:str):
    if(line != ''):
        line = line.strip()
        # 去除文本中的英文和数字
        line = re.sub("[a-zA-Z0-9]", "", line)
        # 去除文本中的中文符号和英文符号
        line = re.sub("[\s+\.\!\/_,$%^*(+\"\'；：“”．]+|[+——！，。？?、~@#￥%……&*（）]+", "", line)
        return line
    return None

#文本切割
```

```
def sent2word(line):
    segList = jieba.cut(line,cut_all=False)
    segSentence = ''
    for word in segList:
        if word != '\t':
            segSentence += word + " "
    return segSentence.strip()



if __name__ == '__main__':
    data = xlrd.open_workbook('附件3.xlsx')
    table = data.sheet_by_name('Sheet1')
    UrlList = table.col_values(4)
    target = codecs.open('cut.txt', 'w', encoding='utf-8')
    for i in UrlList:
        print(i)
        line = clearTxt(i)
        seg_line = sent2word(line)
        target.writelines(seg_line + '\n')
```

附录 8：

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import pandas as pd

def labels_to_original(labels, forclusterlist):
    assert len(labels) == len(forclusterlist)
    maxlabel = max(labels)
    numberlabel = [i for i in range(0, maxlabel + 1, 1)]
    numberlabel.append(-1)
    result = [[] for i in range(len(numberlabel))]
    for i in range(len(labels)):
        index = numberlabel.index(labels[i])
        result[index].append(forclusterlist[i])
    return result

if __name__ == '__main__':
    # 分类数
    num = 3
```

```
    # 读取语料库
    corpus = []
    txt = open("cut.txt", "r", encoding='utf-8').read().split("\n")
    for str in txt:
        corpus.append(str)

    # 该类会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(max_features=20000)
    # 该类会统计每个词语的 tf-idf 权值
    tf_idf_transformer = TfidfTransformer()
    # 将文本转为词频矩阵并计算 tf-idf
    tfidf =
tf_idf_transformer.fit_transform(vectorizer.fit_transform(corpus))
    # 获取词袋模型中的所有词语
    tfidf_matrix = tfidf.toarray()
    # 获取词袋模型中的所有词语
    word = vectorizer.get_feature_names()
    print(word)
    # # 统计词频
    print(tfidf)

    # 聚成 5 类
    clf = KMeans(n_clusters=num)
    s = clf.fit(tfidf_matrix)

    # 每个样本所属的簇
    label = []
    i = 1
    while i <= len(clf.labels_):
        label.append(clf.labels_[i - 1])
        i = i + 1
    # 获取标签聚类
    y_pred = clf.labels_

    # pca 降维，将数据转换成二维
    pca = PCA(n_components=2)  # 输出两维
    newData = pca.fit_transform(tfidf_matrix)  # 载入 N 维

    xs, ys = newData[:, 0], newData[:, 1]
    # 设置颜色
    cluster_colors = {0: 'r', 1: 'yellow', 2: 'b', 3: 'chartreuse',
4: 'purple', 5: '#FFC0CB', 6: '#6A5ACD',
                      7: '#98FB98'}
```

```
    # 设置类名
    cluster_names = {0: u'类0', 1: u'类1', 2: u'类2', 3: u'类3', 4:
u'类4', 5: u'类5', 6: u'类6', 7: u'类7'}

    df = pd.DataFrame(dict(x=xs, y=ys, label=y_pred, title=corpus))
    groups = df.groupby('label')

    fig, ax = plt.subplots(figsize=(8, 5))  # set size
    ax.margins(0.02)
    for name, group in groups:
        ax.plot(group.x, group.y, marker='o', linestyle='', ms=10,
label=cluster_names[name],
                color=cluster_colors[name], mec='none')
    plt.show()

    res = labels_to_original(y_pred , corpus)
    print(res)
    for i in range(len(res)):
        for j in range(5):
            print(res[i][j])
        print("======================")
```

附录 9：

```
import xlrd
import jieba

data = xlrd.open_workbook('附件 3.xlsx')
stop = open('stop.txt', 'r', encoding='utf-8-sig')
stop = stop.read().split()
table = data.sheet_by_name('Sheet1')
UrlList = table.col_values(2)
cipin = {}
shu = 0
print(len(UrlList))
for i in range(1, len(UrlList)):
   d = []
```

```
    fencis = jieba.lcut(UrlList[i], cut_all = True)
    for fenci in fencis:
        if fenci not in stop:
            if fenci in d:
                continue
            else:
                d.append(fenci)
    print(len(d))
    for a in d:
        cipin[a] = cipin.get(a, 0)+1

print(len(cipin))

with open('cipin3.csv','w',newline='',encoding='utf-8-sig') as f:
    for s in cipin:
        key = s
        value = cipin.get(s)
        hang = s + ',' + str(value) + '\n'
        f.write(hang)
```