“泰迪杯”数据挖掘挑战赛

题 目：基于文本挖掘的智慧政务系统

摘要:

问题 1：基于所研究的问题的实际性，首先对文件进行了一定的预处理，最终从分析 tf-idf 权值向量的方法入手，给出了引入伯努利朴素贝叶斯模型对留言进行一级分类的解决方案，并最后利用 f-score 方法对最终所给出模型的测试效果进行了评价，测试的效果显示较好。

问题 2：在本问题中，首先利用文档向量的夹角余弦相似度量方法对相似的留言主题进行提取和归并，再计算每一类问题的四个指标：总赞同数（社会关注）、时间（问题的时间跨度）、涉及人数（社会广度）、反映次数（紧急程度），利用熵权法综合四个指标计算每一类问题的热度指数，找出排名前 5 的热点问题，并利用 TF-IDF 算法进行关键词提取以描述问题，最后列出热点问题留言明细表。

问题 3：本问题中，我们使用余弦估计、BM25 等算法对答复的相关性进行评价，并利用 TF-IDF 算法提取关键词，计算关键词词频对答复完整性进行评估，在可解释方面我们使用到网络爬虫技术对答复内容进行可靠性研究，以及利用 NLP 自然语言处理算法对问题进行情感分析，来评价答复的质量。

关键词：伯努利朴素贝叶斯算法，余弦相似度算法，熵权法，TF-IDF 算法，BM25 算法

一、 引言

随着大数据时代的到来，互联网技术在我们生活的方方面面都发挥了不可小觑的作用。如何对有限的数据进行量化计算，得到数字化的结果，并进而对未来发展趋势做出一定的预测估计，是我们不断研究努力的方向。

数字化时代使得人们的生活更加便捷。我们渴望将每个具象事物抽象化，进而归纳总结，在节约时间的基础上得到有用的结论。文本挖掘就是一门基于数据挖掘的技术。对于信息化的文字，通过特定的算法，将文字转换为数字信息，并进行数据拟合，得到抽象的结论，将人们从信息翻涌的浪潮中解放出来，提高信息获取的便捷性。对于一篇文章，往往浏览关键词便可得其概要。对于热点问题，人们只需浏览热搜，便可知道问题的重要性与影响力。对于政务处理，人们只需将归纳处理后的留言进行一一回复，便可很好地解决民生问题。

文本挖掘就是这样一门文本处理技术。互联网使得每个个体都有了发言权，每天都会出现无数条评论。如果将之应用到政务处理上，将会使得百姓受益人数大大增加。百姓的反馈是一种自然语言，自然语言处理是实现数字化的第一步。因而我们希望能够掌握相应的文本挖掘的理论知识，将百姓在网上对生活方方面面亟需解答的问题进行归纳统计，并反馈给政府的有关部门。政府也可以通过网络进行线上解答，这也弥补了线下办公周期长、手续繁杂等诸多不便利的缺点。

智慧政务是百姓渴望达到的一种理想问政方式。百姓足不出户，便能很快有效地解决困扰自己已久的问题，部门工作人员无需走访调研，也能实时掌握百姓的思想动态，进而不断完善治理机制，提升服务的质量与水平。因此，对百姓留言的文本挖掘就显得尤为重要。

本项目着重阐述如何利用已有算法对百姓留言进行合理的归类评价，并在此基础上，对政府的答复意见建立一套综合的评价体系。通过对提问与解答两个维度的深层次分析，政府便可以实时掌握民生动态，在问政中不断提升问政服务能力。百姓便能时刻发表自己的意见言论，话语权不断增强，这样能够大大提高政府的问政效率。

一、实验方案

2.1 初步方案设计（解题思路）

问题 1：

本题所需要解决的问题是对群众的留言按照所给定的指标进行一级分类，并对最终的分类结果进行评价。此问题的核心在于要首先通过对已经经过分类的训练数据构造模型，并对机器进行训练。之后，再让机器对实际参与测试的数据进行分类，并通过 f-score 算法来衡量机器分类的效果。

问题 2：

问题 2 具体要解决三个问题：问题识别，即从众多留言中识别出相似的留言；问题分类，即把相似的留言归为同一问题；热度评价，即确立若干评价指标，并应用一定的方法综合各个指标之后排名得出热点问题表。

问题 3：

问题 3 具体需要回答三个方面的问题，即确立相应的评价算法对答复意见的相关性、完整性以及可解释性三个维度进行评估。相关性，即答复意见的内容与问题的相关性；完整性，即是否满足某种规范；可解释性，即答复意见中内容的相关解释。

2.2 方案具体分析

问题 1：

由于分类算法程序的运行必须要考虑算法的运行时间和效率问题，处理大规模的原始数据显然是不明智的，所以首先要对原来的数据文件进行一定的预处理工作以减轻程序运行的负担，将分类标签用数字进行分野，也方便程序的识别。而考虑到通过 jieba 模块可以有效地对中文语句进行分词操作，并且分词后还方便我们对无关词语进行剔除操作，极大提高分类的准确性。所以我们在程序中采取了这样的做法，并且之后还通过设立 TXT 格式的无关词表的方式去除了大量对分类没有贡献的词语。

之后进入分类的环节，由多分类转化为二分类的思想出发，可以采用在分类问题中比较有效的的伯努利贝叶斯模型，先分析得到相同分类标签下语句的词频向量，进而得到 TF-IDF 权值向量，交由模型进行训练，最后进行测试，并且可

以将预测结果和测试的原数据使用计算 f-score 值的函数进行运算，最终得到评价指标。解决步骤如下**图 1** 所示。

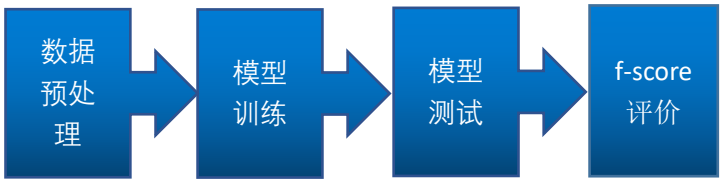

图 1

问题 2：

在问题 2 中，我们首先要解决问题识别，判断各个问题的相似度，由于留言详情字数较多而不突出问题重点，我们选择语句凝练的留言主题进行问题相似度识别，开始我们使用的是备受好评的 Simhash 算法，后来发现用 Simhash 算法计算部分相似的留言时，海明距离数值实在过大，不符合相似的文本海明距离一般小于等于 3 的条件。后来经过网上查阅后发现 Simhash 算法不适合短文本相似度的判断。所以我们更换了一种相似度判断方法：余弦相似度算法。在对相似文本判断它们的余弦相似度时，我们发现夹角余弦普遍大于 0.5，说明该方案可行，因此我们实现问题识别时采用余弦相似度算法。

问题分类中，我们要把两两相似的问题整合到同一类问题中，在这里运用对列表的各种操作，在这里不加以赘述。

热度评价过程中，我们首先先确立可量化可用来判断热度的指标，表格中可以得到点赞数和反对数以及留言时间，于是我们用两者之差来衡量社会关注尺度，每类问题的时间跨度越大，就说明该问题长时间未得到解决，越需要政府的关注，因此也可以作为评价热度的一个时间指标。一类问题提出的次数越多，代表该问题越紧急，可以作为衡量紧急程度的一个热度评价指标。由于一个人可以多次反映相同问题，反映次数并不完全代表着反映人数，将反映人数作为热度评价的社会广度指标。四项指标如**图 2** 所示。

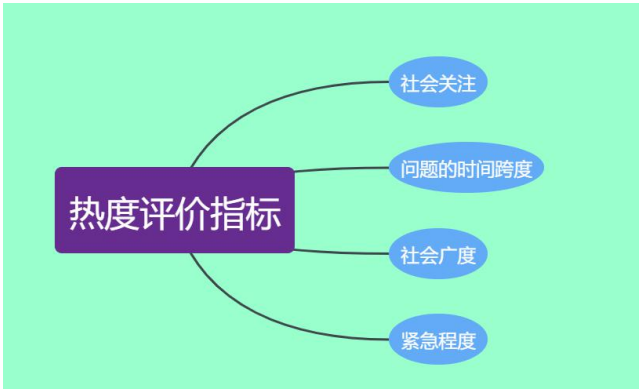

图 2

在对每项热度评价指标进行量化后，如何综合各个指标得到每类问题的热度呢？我们当然要对对各项指标占比进行计算，这里采用熵权法计算各项指标的重要程度，最后利用各项指标占比乘以指标量化后结果计算求和得到每类问题的热度并进行排名。

最后利用提取出热度排名前五的问题进行主题提取并绘制出热点问题表以及热点问题留言明细表。

问题 3：

在问题 3 中，首先，我们对文本数据进行一些前期的预处理工作。包括去停用词、使用结巴分词等等。

针对答复意见的相关性，我们将这里的相关性理解为答复意见与留言主题以及留言详情二者的相关性。因此我们想到使用基于文本相似度与相关性的余弦相似度算法、BM25 算法以及 TF-IDF 算法来具体实现。BM25 算法常用来对算法进行评分。通过对文本的每个查询词进行语素解析，并将其与检索查询结果进行加权，求和得出其相关性的得分。而余弦相似度则从另一层面进一步确定文本之间的相关性程度，并将其量化为百分比形式。

对于答复意见的完整性，我们考虑使用统计特定关键词词频的方法来评估。通过观察数据表格我们发现，一个完整的答复意见往往包括开头问候、中间主体解决问题以及结尾致谢三部分构成。而每个答复意见主体解决问题一般都不会缺失，故而我们只需要对开头结尾的句子进行关键词提取，再计算其出现频率，并将之与答复条数进行比对，从而掌握整个数据表格中答复语言结构的完整程度。

最后，对于答复意见的可解释性，我们有以下三个角度的思考。首先，我们考虑到，一个答复意见可解释性很大程度上取决于人们现有的认知程度。因而，如果答复意见中包含有相关法律法规，往往能够使人们一目了然，且有理有据，不会超出百姓的认知水平。因此，我们查阅了相关网站（中国政府网等），并利用 Python 爬虫技术对相关网页进行内容的提取，并转化为 txt 文档，再利用前一问相关性的评估方法，对答复内容与相关法律法规条目的契合程度进行量化评估。得到了法律法规依据层面的评价标准。其次，答复意见的可解释性与答复语言的感情色彩也有一定的关系。随着国家倡导服务型政府的建设，“微笑服务”逐渐成为政府办公的重点之一。一个正面感情倾向的答复意见往往能够增强其自身的可读性，使人们心情愉悦从而更愿意听取相关部门的意见。因此，我们使用了 NLP（自然语言处理技术）来对答复意见的语句进行情感分析。在具体分析中，我们会使用朴素贝叶斯算法，对相关单词的正面情感频率以及负面情感频率进行计算拟合，归纳出每条语句的感情倾向。最后，评价答复意见的可解释强弱还需后期对市民的走访调查，了解其采用意见的程度来评价答复意见的有效性。

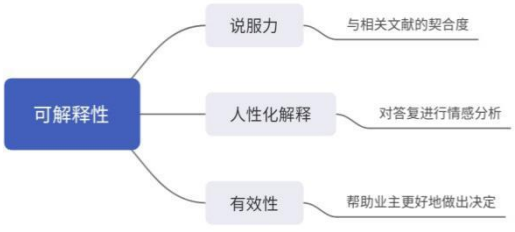

图 3

三、 实验过程

3.1 前期准备

在进行具体的数据挖掘工作之前，我们首先详细地补充学习了 Python 语言的高阶技巧，掌握牢固的 Python 编程基础。并且结合题目的需要，利用现有数据资源并参照详细示例进行深入学习、熟悉利用 Python 语言进行数据挖掘和数据处理的基本步骤、大致流程和基本方法论。在实际编程前，我们还进行了合作探讨，相互交换对于题目含义的意见和看法，明确了各自的编程任务，并且商定了任务的实施细节和算法实施方式，制定了编程安排表。

3.2 方案形成

问题 1：

首先对数据进行预处理。详细的预处理步骤有：读入数据，对数据进行分词和无用词剔除，进而组织形成一份经过简化处理之后的数据，再交给模型进行操作。

首先是数据的读取工作，由前文所述，在读取数据之前，可以先对文件就进行预处理工作。首先我们可以先提前阅读要导入的 xlsx 文件，发现要进行的分类主要是城乡建设、环境保护、交通运输、教育文体、劳动和社会保障、商贸旅游、卫生计生七个大类（一级标签），因此可以进行简化，可以使用 Excel 的查找替换功能修改成对应编号 1~7，这样的简化对后面机器进行更深层次的数据处理和模型训练都有很大帮助。之后回到程序，要先导入用于处理数据分析人物的 pandas 模块，以及解决中文分词问题的 jieba 模块。之后再将要进行处理的附件 2 数据放入项目文件夹中，改名为 samplemax.xlsx，再使用 pandas 模块中的 read_excel()函数读入，由于我们要研究如何对留言主题进行分类，所以我们先把留言主题找出，并剔除留言内容相重复的内容，可以使用系统中带有的 drop_duplicates()函数实现这一点。

关于剔除无用词汇、词语、标点符号，指对于分类没有帮助的包括标点符号和词汇，比如敬语（您好，请），主语（你我他），和语气助词（啊，哦）等等，不含有任何与分类有关的性质，都可以剔除掉。由于无用词汇浩如烟海，所以作者特意挑选了一份在教学案例中用于剔除无用词汇的 TXT stopword 文本，将其命名为 unworked.txt，并导入程序，将无用词语表整体设置为列表形式。

之后，出于程序运行时间的考虑，调用匿名函数对导入的**留言主题**而非留言详情进行分词（使用 jieba.lcut()函数），再对分词之后的留言主题进行无用词汇剔除的工作。进行上两个部分的操作之后，再对数据进行整合，加入其他的列，形成一个完整的.csv 文件导出，以便根据此进行后续程序的改进。至此，初步的数据预处理的就完成了。接着，再对上述程序进行函数封装，改造为一个可以移植和导入的函数 file_input()。

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 15 14:02:23 2020

@author: 30543
"""
import pandas as pd
import jieba

def file_input(file='samplemax.xlsx'):
    data1=pd.read_excel(file,Header=None)

    data_dup=data1['留言主题'].drop_duplicates()
    a=pd.read_csv('unworked.txt',encoding='GB18030', sep='AK47', header=None,engine='python')
    a=list(a.iloc[:, 0])

    data_cut=data_dup.apply(lambda x:jieba.lcut(x))
    stopwords=list(',, ')+a
    data_after_step=data_cut.apply(lambda x:[i for i in x if i not in stopwords])
    labels=data1.loc[data_after_step.index,'一级标签']

    adata = data_after_step.apply(lambda x: ''.join(x))
    adata2=data1['留言编号']
    bdata=data1['留言用户']
    cdata=data_after_step[:]
    ddata=data1['留言时间']
    edata=data1['一级标签']
    data_new=pd.concat([adata2,bdata,cdata,ddata,edata],axis=1)

    data_new.to_csv('samplenew.csv')

    return adata, data_after_step, labels,data_new

```

图 4：封装好的 file_input()函数

接着，将进入更深一级的数据预处理和模型的训练过程。写一份新的程序，首先要在新的程序中导入 pandas 模块、先前编写的 file_input()函数，以及从机器学习 sklearn 模块中导入用于构建伯努利贝叶斯模型算法的 BernoulliNB 模块，用于切分训练数据和测试数据的 train_test_split 模块，用于计算词频向量和 tf-idf 权值向量的模块 CountVectorizer 和 TfidfTransformer，以及用于计算最后 f-score 值的 f1_score()函数。

由于原先的数据排布十分整齐，各个类的问题集中分布的在一起，并且在此基础上要将问题分为七类，为了简化问题，同时提升训练的效果，可以尝试将七分类问题转化为七个二分类问题，即每一次分类都将属于本类的对象类别（一级标签）设为“1”，不属于本类的分类对象设为“0”，同时还要在打乱数据之后进行训练，这样也可以体现程序的适应性和有效性，故在本程序中引入了 for 循环结构。

由于我们要对问题的主题和问题的分类（即一级标签）进行匹配（即交由模型进行训练），所以我们要在数据中导出“留言主题”和“一级标签”两类，来划分训练集和测试集(比例为 0.8：0.2)，并计算词频向量和 tf-idf 权值向量。到这里，整个数据预处理的过程才完全结束。

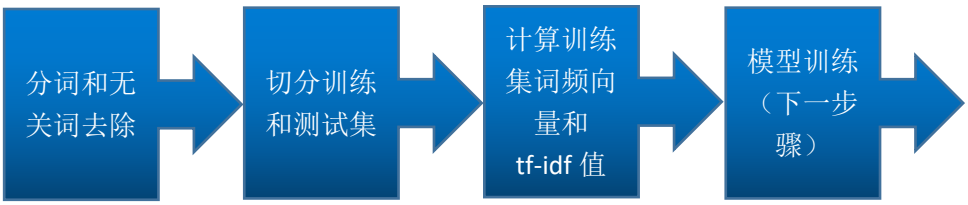

图 5：数据预处理的步骤

接下来是模型训练和测试结果的给出过程，首先导入伯努利贝叶斯模型，调用 model.fit()函数进行训练，之后再由 predict 函数操作测试数据给出预测结果，之后再与标准结果进行 f_score 运算（micro 模式），给出最终的 f_score 评价结果，利用循环结构还可以求出均值，这就是所要求的 f_score 的最终值，最终的结果位于 0.85-0.86 左右，说明此模型应该是比较良好的模型。
f-score 方法的计算公式：

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

问题 2：
数据预处理：
在利用余弦相似度算法之前，要首先对读取出的文本进行分词处理，我们采用 jieba 分词的默认精确模式，即试图将句子最精确地切开。值得一提的是，这里我们没有对文本进行去停用词处理，这是因为处理的文本——附件 3 中的留言主题能较为精确地概括问题而没有冗余，因此我们无需做停用词处理。
相似度计算：
我们采用余弦相似度量来计算个体之间的相似程度，具体步骤如图 3 所示：

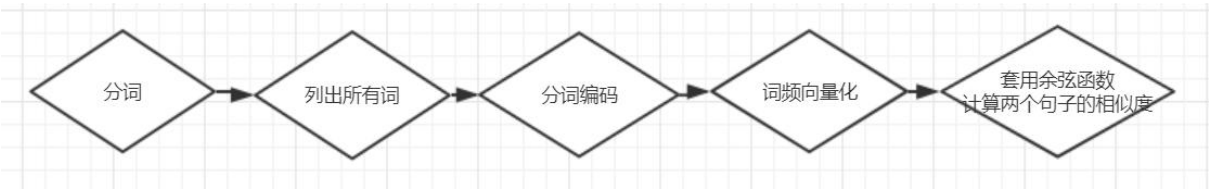

图 6：相似度计算过程

用一个向量空间中两向量夹角的余弦值来表征个体之间是否相似，余弦值越接近 1，夹角越趋近于 0，则表示文本越相似，反之差异越大。
在这里我们封装了一个计算相似度的函数 Similarity[1]，具体算法如**图 7** 所示：

```
def Similarity(s1,s2):
    s1_cut = [i for i in jieba.cut(s1, cut_all=True) if i != '']
    s2_cut = [i for i in jieba.cut(s2, cut_all=True) if i != '']
#     jieba.analyse.set_stop_words('stopword.txt')
    word_set = set(s1_cut).union(set(s2_cut))
    word_dict = dict()
    i = 0
    for word in word_set:
        word_dict[word] = i
        i += 1
    s1_cut_code = [word_dict[word] for word in s1_cut]
    s1_cut_code = [0]*len(word_dict)
    for word in s1_cut:
        s1_cut_code[word_dict[word]]+=1
    s2_cut_code = [word_dict[word] for word in s2_cut]
    s2_cut_code = [0]*len(word_dict)
    for word in s2_cut:
        s2_cut_code[word_dict[word]]+=1
     # 计算余弦相似度
    sum = 0
    sq1 = 0
    sq2 = 0
    for i in range(len(s1_cut_code)):
        sum += s1_cut_code[i] * s2_cut_code[i]
        sq1 += pow(s1_cut_code[i], 2)
        sq2 += pow(s2_cut_code[i], 2)
        try:
            result = round(float(sum) / (math.sqrt(sq1) * math.sqrt(sq2)), 2)
        except ZeroDivisionError:
            result = 0.0
    return result
```

图 7

问题分类：

将夹角余弦值大于 0.5 的问题判定为相似，即为同一类，得出若干个两两相似问题的下标，再将这些元素纳入同一个列表 list 中，将两两相似问题的前一个问题的下标作为索引 index，将与 index 对应留言主题相似的若干留言主题归并到同一列表中，由于有多个 index，故有多个列表，将这些列表作为 list 型元素统一归并到一个列表 all 当中去。由于考虑不同问题有不同表述，造成 index 不能完全找出对应相似的问题，故需要通过与 index 相似问题的余弦值大于 0.5 的问题，到此为止，可以将同一类问题的下标归并到同一列表中并进行去重，即可完成分类。整个过程可能比较繁琐，但也是必要的，具体算法如图 8 所示：

```
list=[]
for i in range(4326):
    for j in range(i):
        if(Similarity(cols[i],cols[j])>0.5):
            print(i,j)
            list.append(i)
            list.append(j)

index_list_repeat=[]
for i in range(len(list)):
    if(i%2==0):
        index_list_repeat.append(list[i])
index_list=[]
for i in range(len(index_list_repeat)):
    if index_list_repeat[i] not in index_list:
        index_list.append(index_list_repeat[i])
def search(x):
    depart=[x]
    for i in range(len(list)):
        if list[i]==x:
            depart.append(list[i+1])
    return depart
all=[]
for i in range(len(index_list)):
    all.append(search(index_list[i]))
print(all)

for i in range(len(all)):
    for j in range(i):
        try:
            a = [x for x in all[i] if x in all[j]]
            b = [y for y in (all[i] + all[j]) if y not in a]
            if len(a)>0:
                all[i]=a+b
                all.pop(j)
        except:
            continue
print(all)
```

图 8

得出的 all 列表由于过长，在这里就不展示了。最后总共得出的类别（即 all 的 列表元素个数）有 630 类（个别特殊问题除外）

指标计算：

这里采用了 4 个指标，即总赞同数（社会关注）、时间（问题的时间跨度）、涉及人数（社会广度）、反映次数（紧急程度）。

总赞同数=总点赞数-总反对数

时间跨度用 excel 中表示日期的数值差来表示。首先将日期统一转化为 yyyy:mm:dd hh:mm:ss 的日期格式。时间跨度越大，代表该问题长时间未得到解决，因此就越重要。

特别说明：在表格的 E 列插入的一列为留言时间（文本格式），因此列标与之前

不同。
将得到的各个指标写入热度评价指标 real.xlsx 中，得到的表格部分如图 9 所示：

问题ID	总点赞数	反映次数	反映人数	问题紧迫性
1	1	2	1	0.00331
2	1	3	3	145.5169
3	0	2	2	0.71397
4	18	2	1	24.82726
5	-1	2	1	0.006678
6	3	2	2	5.406609
7	0	2	2	2.062546
8	0	2	1	22.34061
9	5	3	1	92.05361
10	3	2	2	293.3646
11	3	2	2	56.39853
12	0	2	1	0.01419
13	50	2	2	1.193762
14	2	3	1	294.127
15	0	2	2	31.75726
16	2	2	1	5.978322
17	2	2	2	61.48797
18	2099	2	2	105.9036
19	4	2	1	0.002836
20	54	2	2	4.532454
21	0	2	2	191.0526
22	7	3	3	154.4565
23	0	2	2	332.9372
24	1	2	2	277.981

图 9

热度评价：
利用熵权法，计算各个指标对于热度指数贡献的比重，从而加权得出每一类问题的热度指数。具体步骤如图 10 所示：

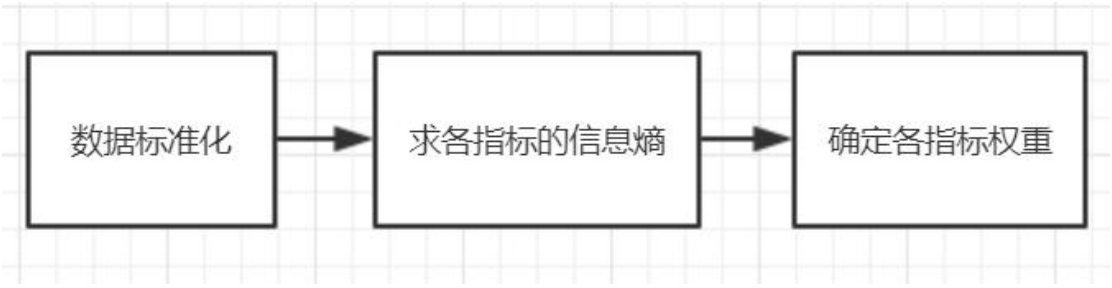

图 10

算法以及得出的各指标权重[2]如图 11 所示：

```
import numpy as np #导入数值计算拓展模块
import pandas as pd #导入数据分析模块
from sklearn import preprocessing
data=pd.read_excel(r"C:\Users\ASUS\Desktop\建模\示例数据\热度评价指标real.xlsx",sheename="Sheet1",index_col="问题ID") #读取数据
zdata=preprocessing.MinMaxScaler().fit_transform(data) #极大极小标准化
def entropy(data): #定义熵函数,返回综合得分
    m,n=np.shape(data) #数据维度
    data[np.where(data==0)]=0.0001 #替换0值
    data=pd.DataFrame(data).values #数据框化后矩阵化
    col_sum=data.sum(axis=0) #求列和
    pij=data/col_sum #占比
    entrop=-np.sum(pij*np.log(pij),axis=0) #信息熵
    w=(1-entrop)/np.sum(1-entrop,axis=0) #权重
    print(w) #观察权重是否合理
    score=np.dot(data,w) #得分
    return score
if __name__=="__main__":
    score=pd.DataFrame(entropy(zdata),index=data.index) #得分
    score.columns=["综合得分"] #给出列名
    result=pd.concat([data,score],axis=1) #沿着列的方向水平延伸合并
    #print(result) #测试
    result.to_excel("output.xlsx") #保存为excel
```

```
[0.2614194  0.21881327 0.24016828 0.27959905]
```

图 11

再利用 excel 里的 f(x)操作将各类问题的热度计算出来，如图 12 所示：

F631 =0.2614194*B631+0.21881327*C631+0.24016828*D631+0.27959905*E631

问题ID	总点赞数	反映次数	反映人数	问题紧迫性	热度指数
1	1	2	1	0.00331	0.940139745
2	1	3	3	145.5169	42.32476024
3	0	2	2	0.71397	1.117588408
4	18	2	1	24.82726	12.32502148
5	-1	2	1	0.006678	0.41824265
6	3	2	2	5.406609	3.213903983
7	0	2	2	2.062546	1.494649085
8	0	2	1	22.34061	6.92420911
9	5	3	1	92.05361	27.94180731
10	3	2	2	293.3646	83.72669305
11	3	2	2	56.39853	17.47119674
12	0	2	1	0.01419	0.681762279
13	50	2	2	1.193762	14.3227077
14	2	3	1	294.127	83.65709026
15	0	2	2	31.75726	9.797261972
16	2	2	1	5.978322	2.872166704
17	2	2	2	61.48797	18.63278117
18	2099	2	2	105.9036	579.2478392
19	4	2	1	0.002836	1.724265265
20	54	2	2	4.532454	16.30188045
21	0	2	2	191.0526	54.33609296
22	7	3	3	154.4565	46.3927627
23	0	2	2	332.9372	94.00688443
24	1	2	2	277.981	78.90259825

图 12

热度排名：

找出热度指数排名前 5 的问题类别 ID 和对应在附件 3 中的行号，得出结果如图 13 所示

```
问题类别ID为
18
对应表格中的行为
[845, 820]
问题类别ID为
43
对应表格中的行为
[1385, 251]
问题类别ID为
295
对应表格中的行为
[3194, 156]
问题类别ID为
572
对应表格中的行为
[2414, 4206, 2003, 2459, 2570, 3425, 4136, 3187, 3857, 3941, 471, 645, 1156, 1516, 3281, 3885, 394, 1355, 1401,
1636, 1862, 1994, 2009, 2061, 2102, 2384, 2544, 2741, 2966, 3013, 3084, 3182, 3270, 3419, 3422, 3712, 793, 3485,
739, 225, 336, 1485, 2160, 3377, 473, 1109, 1580, 1609, 1753, 2375, 2756, 3000, 3145]
问题类别ID为
630
对应表格中的行为
[1914, 4324, 4325, 4326, 4327, 305, 2843, 3314, 4323, 3796, 2265, 1989]
```

图 13

该结果未按热度排名列出

问题 3：

首先是一些准备工作，我们通过调用测试文本，将 Excel 表格里面答复意见、留言主题、留言详情三列的内容分别转换成了三个 txt 文档。接着我们需要定义一个 stopword.txt 文档来对文本内容进行去停词操作，同时我们还需要对句子进行分词操作。具体执行见图 14、15、16 所示。

```
print(f)
```

```
[{'': 1, '现将': 1, '网友': 2, '平台': 1, '问政': 1, '西地省': 1, '栏目': 1, '胡华衡': 1, '书记': 1, '留言': 1, 'A2': 1, '区景蓉': 1, '花苑': 1, '物业管理': 1, '调查核实': 1, '情况': 1, '答复': 1, '您好': 1}, {'感谢您': 1, '工作': 1, '信任': 1, '支持': 1}, {'平台': 1, '栏目': 1, '胡华衡': 1, '书记': 1, '留言': 1}, {'A2': 1, '区景蓉': 1, '花苑': 1, '物业管理': 1, '情况': 1, '已': 1, '收悉': 1}, {'现将': 1, '调查': 2, '情况': 1, '答复': 1}, {'针对': 1, '来信': 1, '小区': 1, '停车': 1, '收费': 1}, {'景蓉华苑': 1, '业委会': 1, '2019': 1, '年': 1, '4': 2, '月': 2, '10': 1, '日至': 1, '27': 1, '日以': 1, '意见': 1, '收集': 1, '方式': 1, '业主大会': 1}, {'业委会': 1, '统计': 1}, {'超过': 1, '三分之二': 1, '业主': 1, '同意': 1, '收取': 1, '停车': 1, '管理费': 1}, {'业主大会': 1, '结束': 1, '后': 1, '业委会': 1, '业主': 1, '提出': 1, '意见': 1, '建议': 1, '疏理': 1, '归纳': 1, '反馈': 1}, {'业委会': 1, '制定': 1, '停车': 1, '收费': 1, '标准': 1, '不': 1, '高于': 1, '周边': 1, '小区': 1, '价格': 1}, {'针对': 1, '来信': 1, '物业公司': 1, '去留': 1}, {'5': 2, '月': 1, '日': 1, '下午': 1, '辖区': 1, '桂花': 1, '坪': 1, '街道': 1, '牵头': 1, '组织': 1, '社区': 1, '物业公司': 1, '业主': 2, '委员会': 1, '代表': 1, '会议': 1}, {'区': 1, '住房': 1, '城乡': 1, '建设局': 1, '参加': 1, '会议': 1}, {'综合': 1, '意见': 1, '后': 1}, {'辖区': 1, '桂花': 1, '坪': 1, '街道': 1, '区': 1, '住房': 1, '城乡': 1, '建设局': 1, '已': 1, '业委会': 1, '依法': 1, '依规': 1, '业主大会': 1}, {'业主大会': 1, '表决': 1, '再': 1, '执行': 1, '程序': 1}, {'再次': 1, '感谢您': 1, '我区': 1, '工作': 1, '理解': 1, '关心': 1}, {'2019': 1, '年': 1, '5': 1, '月': 1, '9': 1, '日': 1, '': 1}]
```

图 14

```
print(tf)
```

```
{'': 2, '现将': 2, '网友': 1, '平台': 2, '问政': 1, '西地省': 1, '栏目': 2, '胡华衡': 2, '书记': 2, '留言': 2, 'A2': 2, '区景蓉': 2, '花苑': 2, '物业管理': 2, '调查核实': 1, '情况': 3, '答复': 2, '您好': 1, '感谢您': 2, '工作': 2, '信任': 1, '支持': 1, '已': 2, '收悉': 1, '调查': 1, '针对': 2, '来信': 2, '小区': 2, '停车': 3, '收费': 2, '景蓉华苑': 1, '业委会': 5, '2019': 2, '年': 2, '4': 1, '月': 3, '10': 1, '日至': 1, '27': 1, '日以': 1, '意见': 3, '收集': 1, '方式': 1, '业主大会': 4, '统计': 1, '超过': 1, '三分之二': 1, '业主': 3, '同意': 1, '收取': 1, '管理费': 1, '结束': 1, '后': 2, '提出': 1, '建议': 1, '疏理': 1, '归纳': 1, '反馈': 1, '制定': 1, '标准': 1, '不': 1, '高于': 1, '周边': 1, '价格': 1, '物业公司': 2, '去留': 1, '5': 2, '日': 2, '下午': 1, '辖区': 2, '桂花': 2, '坪': 2, '街道': 2, '牵头': 1, '组织': 1, '社区': 1, '委员会': 1, '代表': 1, '会议': 2, '区': 2, '住房': 2, '城乡': 2, '建设局': 2, '参加': 1, '综合': 1, '依法': 1, '依规': 1, '表决': 1, '再': 1, '执行': 1, '程序': 1, '再次': 1, '我区': 1, '理解': 1, '关心': 1, '9': 1}
```

图 15

```
sents = get_sentences(text)
doc = []
for sent in sents:
    words = list(jieba.cut(sent))
    words = filter_stop(words)
    doc.append(words)
print(doc)
document = doc
```

图 16

接着，便是将答复意见与留言主题进行相关性比较。BM25 算法的公式为：

$$Score(Q,d) = \sum_{i}^{n} W_i \cdot R(qi,d)$$

通过查阅资料，我们得知 Q 即 Query, 顾名思义，其表示查询命令。BM25 算法对 Query 的每个词（语素）进行解析，并用 Wi 表示其权重。d 代表检索文档，R(qi,d)则代表语素与文档相关性的量化。

同时，我们采用余弦相似度对语句进行比对，计算两个变量之间的相关性程度。

$$\cos(\theta) = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}} = \frac{X \cdot Y}{|X| \times |Y|}$$

其中，（X,Y 为答复意见与留言主题两个变量），其相似度为 33.50%。[4]

```
# 测试
if __name__ == '__main__':
    with open('D:/taidi/dafu.txt', 'r') as x, open('D:/taidi/zhuti.txt', 'r') as y:
        content_x = x.read()
        content_y = y.read()
        similarity = CosineSimilarity(content_x, content_y)
        similarity = similarity.main()
        print('相关性: %.2f%%' % (similarity*100))
```

```
相关性: 33.50%
```

图 17

为了得到更准确的评价结果，我们还采用了相同的算法对答复意见与留言详情进行了相关度计算，实现如图所示。其相似度为 53.50%。

```
# 测试
if __name__ == '__main__':
    with open('D:/taidi/dafu.txt', 'r') as x, open('D:/taidi/xiangqing.txt', 'r') as y:
        content_x = x.read()
        content_y = y.read()
        similarity = CosineSimilarity(content_x, content_y)
        similarity = similarity.main()
        print('相关性: %.2f%%' % (similarity*100))
```

```
相关性: 53.50%
```

图 18

可以看出，答复意见与留言详情的相关度比较高，这反映了有关部门对留言内容的重视程度在不断加深。

针对答复意见的完整性，我们考虑计算前言、结束语的关键词的词频来进行量化比较。[3]

```
import jieba
import jieba
excludes = {"2019","2018","2017"}
txt = open("D:/taidi/dafu.txt", "r", encoding="gbk").read()
words = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1:
        continue
    else:
        rword = word
    counts[rword] = counts.get(rword, 0) + 1
for word in excludes:
    del(counts[word])
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(33):
    word, count=items[i]
    print((word),count)
```

图 19

```
问题 3075
进行 3055
工作 2750
情况 2524
您好 2395
网友 2162
回复 2054
反映 2053
我们 1859
收悉 1840
相关 1798
建设 1660
感谢您 1564
关于 1460
支持 1452
2018 1345
有关 1326
2019 1252
项目 1233
留言 1231
根据 1225
要求 1114
我局 1001
调查 949
现将 948
目前 948
管理 935
公司 917
学校 917
2017 913
规定 901
部门 891
办理 875
```

图 20

可以看出，“您好，首先感谢您对我们工作的信任和支持”、“再次感谢您对我区工作的理解和关心”、“针对您反映……的问题”类似的前言、结语的关键词的词频都高居榜首，这反映了答复意见在语句结构上的完整性。

接下来，我们要对答复意见进行可解释性分析。通过网络爬虫，我们将收集得到的网页内容词条进行筛选，存储在 1.txt 文件中，

```
import requests
import bs4
#要抓取的目标页码地址
url = 'https://www.baidu.com/link?url=jv5h-npdj9IWME__mjhPeeYFHbhP8jmdm7SRYyYyPAkpViXriuvknpXwWNWfeY05&wd=&eqid=a5d2f145000117df000000045eb39e
#抓取页码内容，返回响应对象
response = requests.get(url)
#查看响应状态码
status_code = response.status_code
#使用BeautifulSoup解析代码,并锁定页码指定标签内容
content = bs4.BeautifulSoup(response.content.decode("utf-8"), "lxml")
element = content.find_all(id='book')
print(status_code)
print(element)
```

图 21

```
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
        社会救助暂行办法 （国务院令第649号） (2014-02-27)
        中华人民共和国保守国家秘密法实施条例 （国务院令第646号） (2014-02-03)
        国务院关于修改部分行政法规的决定 （国务院令第645号） (2013-12-19)
        国务院关于修改《全国年节及纪念日放假办法》决定 （国务院令第644号） (2013-12-11)
        畜禽规模养殖污染防治条例 （国务院令第643号） (2013-11-26)
        国务院关于修改《国际收支统计申报办法》的决定 （国务院令第642号） (2013-11-22)
        城镇排水与污水处理条例 （国务院令第641号） (2013-10-16)
        长江三峡水利枢纽安全保卫条例 （国务院令第640号） (2013-09-16)
        铁路安全管理条例 （国务院令第639号） (2013-09-06)
        国务院关于废止和修改部分行政法规的决定 （国务院令第638号） (2013-07-26)
        国务院关于修改《计算机软件保护条例》的决定 （国务院令第632号） (2013-02-08)
        征信业管理条例 （国务院令第631号） (2013-01-29)
        国务院关于修改《机动车交通事故责任强制保险条例》的决定 （国务院令第630号） (2012-12-27)
        农业保险条例 （国务院令第629号） (2012-11-16)
        国务院关于修改和废止部分行政法规的决定 （国务院令第628号） (2012-11-16)
        国务院关于修改《期货交易管理条例》的决定 （国务院令第627号） (2012-11-05)
        缺陷汽车产品召回管理条例 （国务院令第626号） (2012-10-30)
        国内水路运输管理条例 （国务院令第625号） (2012-10-22)
        教育督导条例 （国务院令第624号） (2012-09-17)
```

图 22

紧接着我们就将文本内容与答复意见进行相关性计算，得出计算结果如图 23 所示。

```
if __name__ == '__main__':
    with open('D:/taidi/1.txt','r',encoding='gb18030',errors='ignore') as x, open('D:/taidi/dafu.txt', 'r',encoding='gb18030',errors='ignore')
        content_x = x.read()
        content_y = y.read()
        similarity = CosineSimilarity(content_x, content_y)
        similarity = similarity.main()
        print('相关性: %.2f%%' % (similarity*100))
```

```
相关性: 13.39%
```

图 23

可以看出，答复意见中引用相关法律法规的频率还是相对比较高的，这从客观层面反映了答复意见的有理有据，符合百姓的认知水平。

在情感色彩处理方面，首先，我们需要使用 BeautifulSoup 模块对文本内容进行清洗，对自然语言处理技术有一个基础的认知。

根据图像我们可以知道，常见的处理技术是 PHP（超文本预处理）技术，我们将一些常见的词汇进行了一个引用频率的拟合，执行如图 24、25、26、27、28 所示，了解了常用自然词汇的使用频率。

图 24

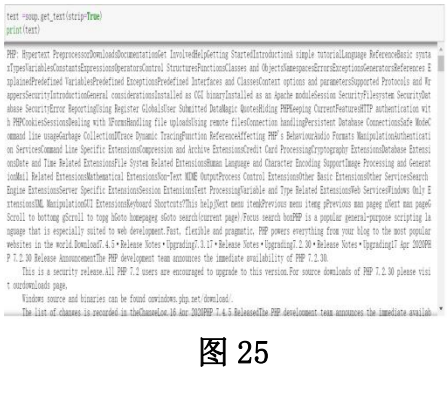

图 25

图 26

图 27

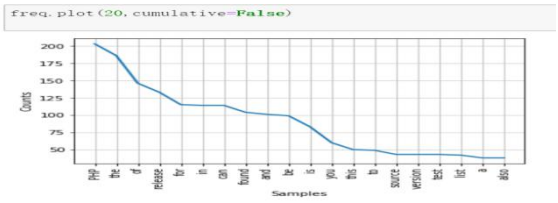

图 28

在掌握了一定的自然语言处理的基础知识之后，我们便开始对答复内容进行情感分析。在这里，我们想到使用朴素贝叶斯分类的方法，对情感倾向进行分析。在算法中，我们将情感倾向分为正向跟负向情感。我们使用的方法是一种很简单的将每条答复进行英文翻译之后，导进我们事先编好的代码语言中，通过对每个单词进行语调概率的分析，计算出每句话的正面或负面情感概率，最终进行综合，评价出每条答复的感情色彩。在下图 29、30、31、32 中，我们以其中一条答复为例，可以看出，其正面情感概率为 0.75，负面情感概率为 0.25，很显然，这是一条正面答复。在我们的调查实验中发现，绝大多数答复都是属于正面情感倾向的答复，这也表明了基层群众自治制度的不断完善，领导层的管理更加注重“微笑服务”，在语言沟通上的能力不断提高。

```
from nltk.tokenize import WordPunctTokenizer
from nltk.probability import FreqDist

neg_str = ['Urban management is necessary. We also hope that managers should enforce the law fairly and civilly ']
pos_str = ['Thank you for your concern, supervision and support for our work thank you for your concern, supervision and support for our work
test_str = 'predictable with no originality'

def count_V(words):
    fdist = FreqDist(words.split())
    tops=fdist.most_common(50)
    return tops

def count_str(words):
    l = words.split()
    return len(l)


if __name__ == '__main__':
    whole_str1 = ''
    whole_str2 = ''
    for words in neg_str:
        whole_str1 += words
    for words in pos_str:
        whole_str2 += words
    V = len(count_V(whole_str1 + ' ' + whole_str2))
    print('V为：%d'%V)
    n_neg = count_str(whole_str1)
    n_pos = count_str(whole_str2)
```

图 29

```
    n_pos = count_str(whole_str2)
    print('n- 为：%d'%n_neg)
    print('n+ 为：%d'%n_pos)
    p_neg = len(neg_str) / (len(neg_str)+len(pos_str))
    p_pos = len(pos_str) / (len(neg_str)+len(pos_str))
    print('P(-) 为：%.6f'%p_neg)
    print('P(+) 为：%.6f'%p_pos)

    arr_neg = count_V(whole_str1)
    arr_pos = count_V(whole_str2)
    dic_neg = {}
    dic_pos = {}
    # 统计每个词语的频率
    for words in arr_neg:
        dic_neg[words[0]]= (words[1]+1) / (n_neg+V)
        dic_pos[words[0]]= (0+1) / (n_pos+V)

    for words in arr_pos:
        dic_pos[words[0]]= (words[1]+1) / (n_pos+V)
        dic_neg[words[0]]= (0+1) / (n_neg+V)

    print('负面答复每个单词的概率：')
    print(dic_neg)
    print('正面答复每个单词的概率：')
    print(dic_pos)

    # 评价测试
    arr_test = count_V(test_str)
```

图 30

```
    # 评价测试
    arr_test = count_V(test_str)
    neg = p_neg;
    pos = p_pos;
    for words in arr_test:
        if words[0] in dic_neg:
            neg *= dic_neg[words[0]]
        if words[0] in dic_pos:
            pos *= dic_pos[words[0]]

    print('负面答复的概率：%.6f'%neg)
    print('正面答复的概率：%.6f'%pos)

    if neg > pos:
        print('所以这是一条负面答复')
    else:
        print('所以这是一条正面答复')
```

图 31

```
V为：33
n- 为：16
n+ 为：34
P(-) 为：0.250000
P(+) 为：0.750000
负面答复每个单词的概率：
{'Urban': 0.04081632653061224, 'management': 0.04081632653061224, 'is': 0.04081632653061224, 'necessary.': 0.04081632653061224, 'We': 0.0408
1632653061224, 'also': 0.04081632653061224, 'hope': 0.04081632653061224, 'that': 0.04081632653061224, 'managers': 0.04081632653061224, 'shou
ld': 0.04081632653061224, 'enforce': 0.04081632653061224, 'the': 0.04081632653061224, 'law': 0.04081632653061224, 'fairly': 0.04081632653061
224, 'and': 0.02040816326530612, 'civilly': 0.04081632653061224, 'for': 0.02040816326530612, 'you': 0.02040816326530612, 'your': 0.020408163
26530612, 'concern,': 0.02040816326530612, 'supervision': 0.02040816326530612, 'support': 0.02040816326530612, 'our': 0.02040816326530612,
'work': 0.02040816326530612, 'thank': 0.02040816326530612, 'Thank': 0.02040816326530612, 'helloFirst': 0.02040816326530612, 'of': 0.02040816
326530612, 'all,': 0.02040816326530612, 'love': 0.02040816326530612, 'Forest': 0.02040816326530612, 'Public': 0.02040816326530612, 'Securit
y': 0.02040816326530612}
正面答复每个单词的概率：
{'Urban': 0.014925373134328358, 'management': 0.014925373134328358, 'is': 0.014925373134328358, 'necessary.': 0.014925373134328358, 'We': 0.
014925373134328358, 'also': 0.014925373134328358, 'hope': 0.014925373134328358, 'that': 0.014925373134328358, 'managers': 0.014925373134328358,
'should': 0.014925373134328358, 'enforce': 0.014925373134328358, 'the': 0.014925373134328358, 'law': 0.014925373134328358, 'fairly': 0.0
14925373134328358, 'and': 0.0447761194029850 7, 'civilly': 0.014925373134328358, 'for': 0.1044776119402985, 'you': 0.05970149253731343, 'you
r': 0.05970149253731343, 'concern,': 0.0447761194029850 7, 'supervision': 0.0447761194029850 7, 'support': 0.0447761194029850 7, 'our': 0.04477
611940298507, 'work': 0.0447761194029850 7, 'thank': 0.0447761194029850 7, 'Thank': 0.029850746268656716, 'helloFirst': 0.029850746268656716,
'of': 0.029850746268656716, 'all,': 0.029850746268656716, 'love': 0.029850746268656716, 'Forest': 0.029850746268656716, 'Public': 0.029850746
6268656716, 'Security': 0.029850746268656716}
负面答复的概率：0.250000
正面答复的概率：0.750000
所以这是一条正面答复
```

图 32

3.3 操作平台：

问题一	Spyder (Python 3.7)
问题二	Jupyter (Python 3.7)
问题三	Jupyter (Python 3.8)

四、实验结果

4.1 实验结论

问题 1：

通过模型对测试集的操作，可以得到 7 个分类的 f-score 值，求其平均值，由于采用了循环计数的结构，因此可以直接算出最终的 f-score 均值为：0.856，多次实验 f-score 值稳定分布在 0.85-0.86 之间。

图 33

问题 2：

得出结果表格：

经过 tf-idf 算法，提取各个留言主题的 6 个关键词并组成句子，即可得到问题描述，对热点问题的时间进行排序即可得到时间范围，最后得到的热点问题表如**图 34** 所示：

热度排名	问题ID	热度指数	时间范围	地点/人群	问题描述
2	18	579.2478392	2019/05/05至2019/08/09	A5区汇金路五矿万金K9县居民	A5区汇金路五矿万金K9县施工方存在问题
3	43	409.4407472	2019/02/31至2019/3/1	A市市民	承办A4区58车贷案警官应关注留言
5	295	179.449073	2019/8/23至2019/09/05	绿地外滩小区居民	A4区绿地海外滩小区距渝长厦高铁太近
1	572	593.3406177	2019/01/01至2019/12/31	A市市民	咨询相关购房政策
4	630	4199881	2017/06/08至2019/11/27	涉外经济学院学生	涉外经济学院强制学生寒假外出实习

图 34

经过图 8 对应的行，经过人工判断是否属于同一类的操作，得到如图所示的热点问题留言明细表：

问题ID	留言编号	留言用户	留言主题	留言时间	留言详情	反对数	点赞数
572	199435	A0002407(	人才购房补助	2019/07/02 13:11:31	领证，为	0	0
572	203760	A0011195	市人才购房补	2019/06/25 15:43:23	社保信息	0	0
572	215591	A0007358	人才新政落户等	2019/03/18 17:35:47	很好的工作	0	0
572	224042	A0001422	及购房补贴实	2019/01/16 11:58:48	单位，201	0	0
572	265577	A0004787	人才购房补贴	2019/12/02 11:57:49	作，年龄	0	1
572	280288	A0004130	购政策的社保缴	2019/08/01 15:05:54	经在A市找	0	0
572	220015	A0007364	购房与购房资	2019/08/09 10:03:41	门就要遭	0	0
572	221141	A0001038	缴纳社保及购	2019/04/08 01:12:00	8月~2018	0	0
572	232575	A0001717	市购房资格审	2019/06/27 11:51:34	在A市已经	0	0
572	235725	A0005360	业安置A市相	2019/07/07 13:41:45	住房，并且	0	0
572	236150	A0004712	口迁移落户的	2019/08/10 09:39:43	雇父母，我	0	0
572	259594	A0001136	二套房购房资	2019/12/15 15:04:13	市处遗办	0	0
572	263429	A0001087	居老人的购房	2019/01/15 21:25:02	有2套房，而	0	0
572	268921	A0006344	住房资格的相	2019/03/06 11:27:56	班时购房	0	2
572	225657	A0005179	人才购房补贴	2019/02/25 14:43:15	在A市安家	2	6
572	229963	A0001046	新政住房补贴	2019/04/24 23:19:34	的是培训合	0	0
572	244951	A0002603	人才租房购房	2019/07/07 19:12:32	就没有补贴	0	0
18	208636	A0007717	五矿万境K9县	2019/08/19 11:34:04	，请问有	0	2097
18	208069	A0009443	开发商与施工	2019/05/05 13:52:50	量是否能符	0	2
43	220711	A0003168	关注A市A4区5	2019/02/21 18:45:14	消息总是	0	821
43	194343	A0001061	贷案警官应跟	2019/03/01 22:12:30	并没有跟进	0	733
630	233759	A909118	经济学院强制	2019/04/28 17:32:51	须去学校安	0	0
630	360111	A1204455	组织学生外出	2019/11/05 10:31:38	上，（晚我	1	0
630	360112	A220235	济学院强制学	2019/04/28 17:32:51	去学校安排	0	0
630	360113	A3352352	学院强制学生	2018/05/17 08:32:04	学校很小	3	0
630	360110	A110021	过年期间组织	2019/11/22 14:42:14	说不是强制	0	0
630	195917	A909119	院组织学生外	2019/11/05 10:31:38	上，（晚我	0	1
630	266368	A0003892	假过年期间组	2019/11/22 14:42:14	说不是强制	0	0
295	191951	A0004144	滩小区距渝长	2019/08/23 14:21:38	是火车，这	0	1
295	263672	A0004144	长赣高铁最近	2019/09/05 13:06:55	我如下问题	0	669

图 35

问题 3：

由于实验结果已在方案形成中给出，这里就不做过多赘述。

4.2 实验不足

(1)本算法循环调用的过程较多，然使模型处理随机数据的能力加强，但是却是以损失模型的运行效率为代价的，这导致了最终运行的时间比较长，内存占用也比较大。

(2)由于中文同义词多，且同一词语在不同语境下有不同含义，由此带来的不确定性使问题 1 和问题 2 的模型精度降低。以问题 2 的模型为例，附件 3 中的“严惩 A 市 58 车贷特大集资诈骗案保护伞”就未能归入“热点问题明细表”中。

五、 实验感想

原来在我们日常交流的语言中，蕴含了这么多奥妙，原来在自然语言处理的发展中，有这么多美妙的算法。在未来，为了更好地让人工智能协助我们生活工作，沟通更加便捷，自然语言处理一定是很重要的模块。但是在执行算法过程中，时间消耗是一个需要克服的弊端。希望在以后的自然语言处理的过程中，改进精度同时减少运行算法的时间。

在我所解决的问题三中，我学会了如何借助 TF-IDF、BM25、余弦估计等算法对文本进行相关性的评估，以及确立了一套完善的答复意见的评价体系。同时对于文本内容的完整性，我初步掌握了如何使用网络爬虫技术来对相关网站内容进行有效提取。

同时，此次大赛还更加向我展示了 Python 作为一门重要编程语言的深刻意义，还激发了我对数据挖掘和数据处理的兴趣，让我了解到了一个更加复杂而有趣的世界，在以后的学习工作和生活中，还可以利用 Python 中的数据处理方法节约时间，让其成为我们的好帮手。

致谢

首先感谢我们的指导老师能够在百忙之中抽空指导我们顺利完成本次赛题的解答以及完善工作，使我们对于数据挖掘的认知有了很大程度的突破。

其次我们还要感谢学校编程协会相关工作人员的指点，使我们在算法掌握程度上有了更大的进步。

同时我们还要感谢本次大赛的组委会，让我们有如此宝贵的机会参与到这次比赛中来，使我们在比赛的过程中，获益良多。

最后感谢本小组每位成员的不懈努力与互帮互助，将我们的比赛成果打造得尽善尽美。

参考文献

[1] https://blog.csdn.net/educationer/article/details/99577712?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-7&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-7

[2] https://blog.csdn.net/starter_____/article/details/82086090?ops_request_misc=%257B%2522request%255Fid%2522%253A%252215855705401919523985926 6%2522%252C%2522scm%2522%253A%252220140713.130056874..%2522%257D&request_id=158557054019195239859266&biz_id=0&utm_source=distribute.pc_search_result.none-task

[3] https://blog.csdn.net/htgt_tuntuntun/article/details/80499427?utm_source=app

[4] https://blog.csdn.net/qq_42280510/article/details/102857696?utm_source=app