第八届“泰迪杯”
全国大学生数据挖掘竞赛

论
文
作
品

作品名称：基于“智慧政务”的文本挖掘及文本分类系统

摘要

文本数据挖掘(Text Mining)技术是指从文本数据中抽取一些有价值的信息和知识的计算机处理技术。

近年来，随着互联网时代的飞速发展，网络留言评论信息呈指数增长。这些文本信息大多表达了用户对于某些事物或现象的看法和论断，对这些数据加以处理和分析，可以较为方便直观地了解人们的想法。基于此，“智慧政务”的概念被提出，微信、微博、市长信箱、阳光热线等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非常大的挑战。与此同时，伴随着着大数据技术、云计算、人工智能领域的发展，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，对于提高政府的管理水平及施政效率具有很大的推动作用。

本文将以处理网络问政平台群众留言为背景，建立一个文本分类器，并定义合理的热度评价指标，将留言进行分类，以便后续将群众留言分派至相应职能部门处理。根据热度评价指标列出某一时间段的热点问题，针对相关部门对留言的答复性意见给出一套评价方案，帮助政府部门更好地实现“智慧政务”数据体系。

关键词：文本挖掘　“智慧政务”　文本分类

ABSTRACT

Text Data Mining is a computer processing technique that extracts valuable information and knowledge from text data.

In recent years, with the rapid development of the Internet era, the information of online comments has increased exponentially. Most of these text messages express the user's views and opinions on certain things or phenomena, and it is convenient to understand people's ideas intuitively by processing and analyzing these data. Based on this, the concept of "smart government affairs" was put forward, WeChat, micro-blogging, mayor's mailbox, sunshine hotline and other online political platforms gradually become the government to understand public opinion, pool ingenuity of people's wisdom, the important channel sedating the people's spirit. The volume of text data related to various social sentiments has been climbing, which has brought great challenges to the work of relevant departments that mainly rely on manual information division and hot-spot sorting. At the same time, with the development of big data technology, cloud computing, artificial intelligence, the establishment of intelligent government system based on natural language processing technology has been a new trend of innovation and development of social governance, which has a great role in improving the level of government management and governance efficiency.

This article will deal with the network political platform mass message as the background, set up a text classifier, and define a reasonable heat evaluation indicators, the message classification, so that the follow-up will be the mass message to the corresponding functional departments to deal with. According to the heat evaluation index listed the hot issues in a certain period of time, for the relevant departments to the reply to the message to give a set of evaluation programs to help government departments to better achieve the "smart government" data system.

Keywords: Text Mining "Smart Government" Text Classification

目录

1.挖掘背景与目标

1.1 挖掘背景

近年来，随着互联网时代的飞速发展，网络留言评论信息呈指数增长。这些文本信息大多表达了用户对于某些事物或现象的看法和论断，对这些数据加以处理和分析，可以较为方便直观地了解人们的想法。基于此，“智慧政务”的概念被提出,微信、微博、市长信箱、阳光热线等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,随着大数据技术、云计算、人工智能领域的发展,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非常大的推动作用。

1.2 挖掘目标

由于基于留言内容的数据挖掘可以帮助政府更好地掌握群众目前生活上的需要与不足。一方面总结并提出基于线性回归的中文文本分类器，并定义合理的热度评价指标,将留言进行分类,以便后续将群众留言分派至相应职能部门处理，根据热度评价指标列出某一时间段的热点问题。从而有效提取群众留言中的有效信息。另一方面通过留言内容与答复详情之间的相似度、政府回复留言的时长以及回复详情的字数酌情进行加权，来对相关部门对留言的答复性意见给出相关评估，帮助政府部门更好地实现“智慧政务”数据体系。

2.问题分析

2.1 问题一的分析

对于问题一，分析附件一和附件二，附件一是总体的大分类，附件二则是题目所要求的一级标签分类所含内容，其中已人为的给出了简化版后的留言主题，对留言主题进行数据预处理，又因为是有标签的，所以排除无监督学习模

型，查找资料后决定结合 tf-idf 对特征进行处理然后结合 Logistics 回归分析模型和支持向量机模型进行拟合。

附件二示例表头

留言编号	留言用户	留言主题	留言时间	留言详情	一级分类
744	A089211	建议增加 A 小区快递柜	2019/10/18 14:44	我们是 A 小区居民...	交通运输

2.2 问题二的分析

对于问题二，要求热点问题表需按照地点或人群进行划分，为便于实现的简便，决定对附件 3 的留言问题进行分词去停止词等一系列的预处理后，对他们直接进行聚类操作，同时为了增强聚类的准确性，决定用 Word2vec 对特征进行表述，聚类完成后在同一类中利用相关处理库对同一问题识别其中的地点或人群，利用 LSI 模型完成对问题的描述，完成要求归类之后再在相同类中对相应的热度指数排序，根据权重定义合理的热度评价指标，然后根据此指标给出对应的热点问题及评价结果。

2.3 问题三的分析

对于问题三，不同的处理者对于答复的评价指标的定义可能会大相径庭。分析附件 4，决定采用政府部门对于留言内容的回复时长作为高效性的评判，答复内容的充实与否即字数的多少作为完整性及服务态度方面的评价，留言内容和答复内容之间的相似度则作为可解释性的判断标准，将三者综合并附上自定义合适的权重进行计算，最后结果即为相对应的答复内容的评价分数。

3.总体流程

由于文本数据的非结构化，以及中文语句结构的复杂化，普通的数据处理方法并不适用于中文文本的处理；本文结合 python 中多个处理中文文本的相关库实现对中文文本数据的处理。总体架构及题目思路流程图如下：

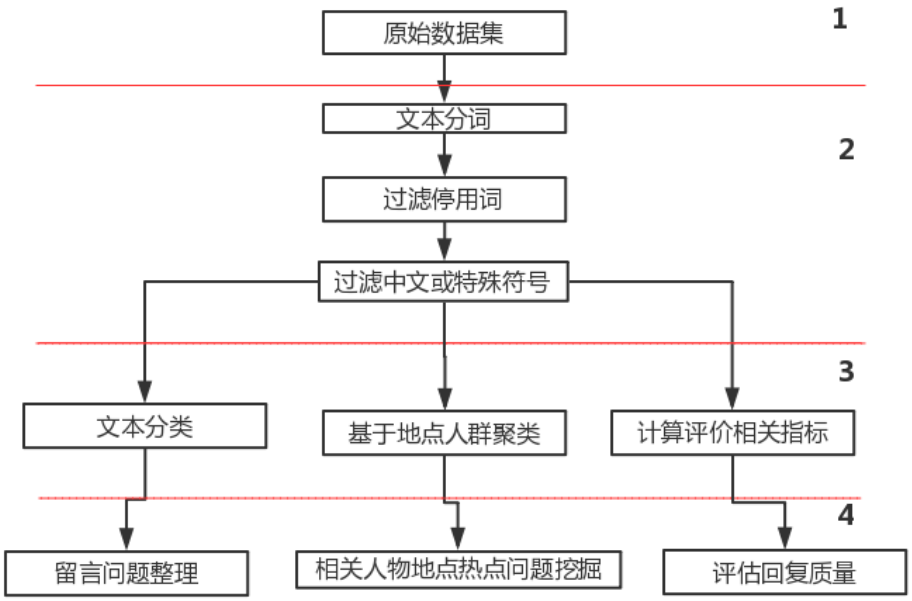

图 3.1　总体流程

第一步：获取数据集，在本文中所有数据集均为已提供的 Excel 表格形式

第二步：对数据集进行一些基本的处理操作,包括了对数据的预处理,并且过滤掉那些没有作用或作用很小的分词、符号以及相关中文分词等；

第三步：相关数据进行处理后，根据不同需要对数据进行特征处理，如同 tfidf 模型向量化或者是利用 Word2vec 对文本进行向量化等操作；

第四步：从相对应的数据挖掘结果中发现对于数据处理有价值的内容。

4. 数据预处理

4.1 中文分词方法概述

中文分词就是一种把连续的字词序列按照一个确定的规范,重新组合成字词序列的过程。中文分词实现是文本挖掘操作的基础，对于输入的一段中文文本,成功的进行中文分词处理,可以达到计算机自动识别所输入语句含义的效果。中文分词技术从属于自然语言处理技术范畴。针对一句完整的话,人们可以通过自己掌握的知识来区分哪些是词,哪些不是词,但在通常情况下计算机无法正确理

解，分词算法就是这个处理过程。由于语言的不同,导致分词的方法和难点也有所不同。对于英语来说,单词之间有空格作为自然分隔符,处理的难点在于大小写代表的不同含义以及符号的用法。而中文由于继承了古代汉语的传统,词语之间没有分隔符。现代汉语的基础表达单元虽然是"词",并且以双字或者多字词为主,可由于人们认识水平有所不同,很难界定词和短语的边界。

根据已知的技术,现有的分词算法可以大致分为三类:基于字符串匹配的分词方法、基于理解的分词方法以及基于统计的分词方法。根据是否与词性标注过程相结合,还可以划分为单纯分词方法和分词和标注相结合的一体化方法。

1.基于字符串匹配的分词方法

基于字符串匹配的分词算法也被称作机械分词算法,一般都需要事先建立足够大的分词词典,即事先的数据库要足够多。然后将自己需要进行分词操作的字符串与之前准备好的分词词典里面的词条一一匹配,假如在词典中可以找到该字符串,就说明匹配成功(可以识别出一个词),那么就把这个字符串当作一个词,从待分词的文本中切分出来,否则不切分。其中,也有三个不同的匹配方法,正向最大匹配法(FMM)、逆向最大匹配法(BMM)和双向最大匹配法,一般逆向最大匹配往往比较准确。

优点：易于理解和实现。

缺点：需要提前准备的足够大的分词词典，且词典的权威一致性无法得到保障；有局限性；耗时较长。

2.基于理解的分词方法

基于理解的中文文本分词方法又被称作知识分词,这种分词方法是让计算机模拟人的思想过程，从而更好地断句，以达到词语识别的要求。它的基本思想就是利用文本语义信息和句法，在分词的同时更好地进行文本识别。通常包括三个部分:分词子系统、句法语义子系统和总控部分。因为它模拟了人对句子的理解过程，所以这种方法将会用到大量的语言知识和信息;难以实现把各种语言信息组织成计算机能够理解并直接读取的形式,因此现期还处于试验阶段。

3. 基于统计的分词方法

统计分词的基本出发点就是把每个词语看做是由一个个字组成的,假如相连的字在不同文本中出现的次数较多,就能够在一定程度上证明这段相连的字有很大可能性是一个词。基于此，字与字相邻出现的概率或频率能够比较好的反映连字成词的可信度。所以可以统计训练文本数据中出现的各个相邻字组合的频数，计算它们之间的相互信息。

根据以上的介绍,本文最终选用了基于语料库的统计分词方法进行文本分词,通过 jieba 库实现分词，实现代码及效果如下图所示：

分词结果

```
def cut(x):
    seg = [jieba.lcut(w) for w in x]
    return seg
text = cut(data.loc[:,"留言主题"].values)#分词列表
cuttext = lambda x:jieba.lcut(x)
cut_text = data.loc[:,"留言主题"].apply(cuttext)
cut_text.head()
```

```
: 0          A 市 经济 学院 体育 学院 变相 强制 实习
  1        A 市 人才 app 上 申请 购房 补贴 通不过
  2              希望 西地省 抗癌 药品 纳入 医保
  3    A5 区 劳动 东路 魅力 城 小区 临街 门面 烧烤 夜宵 摊
  4         请 K3 县 乡村 医生 发 卫生室 执业 许可证
  Name: 留言主题, dtype: object
```

图 4.1.　分词结果

4.2 去停止词

在自然语言中，包括了一些没有实际含义的虚词，比如汉语中的「了、着、吧、啊」和英语中的「the、that、a」等。还有一些结构助词「的、是、对」和

「is、to、on、of」等。不同的应用场景下，对于这些词的处理方式不相同。训练词向量时，由于这些词没有实际意义，应该作为停用词（stop word）去除，并且由于停用词的出现频率很高，停用后能提高检索效率以及准确度。但是在一些查询中，例如 Beatles 乐队的名曲 Let It Be 在去除停用词以后就很难被检索到。因此，在现代搜索引擎中，一般不停用词，而是采用了很多方法避免停用词产生过大开销。

另外，本文中选用了哈工大的停用词表作为去停止词的标准，并且根据附件所给具体内容对停用词表进行了增删的完善，减少停止词带来的误差。

创建停用词列表

```
def stopwordslist():
stopwords=[line.strip()forlineinopen('E:\\stopwords.txt',encoding='UTF-8').readlines()]
return stopwordsstopwords = stopwordslist()
```

4.3 过滤空格及特殊符号

加载数据时发现读取数据后，部分附件的具体内容会多出如下图所示形如'\n'' \t'等符号以及一些标点符号，于是在去停止词之后重新再一次过滤掉空格等特殊的自定义符号。

	留言编号	留言用户	留言主题	留言时间	留言详情	答复意见	答复时间
0	2549	A00045581	A2区景蓉华苑物业管理有问题	2019/4/25 9:32:09	\n\t\t\t\t\t\n\t\t\t\t2019年4月以来，位于A市A2区桂花坪街道...	现将网友在平台《问政西地省》栏目向胡华衡书记留言反映"A2区景蓉花苑物业管理有问题"的调查核...	2019/5/10 14:56:53

图 4.2　内容展示

过滤停用词和特殊符号

```
teshu = ["？ ","； ","： ","。 ","， ","\n","\t"," ","） ","（"] #自定义的符号列表
cut_stop = cut_text.apply(lambda x:[i for i in x if i not in stopwords and i not in teshu])#去停用词
x_cut11 = cut_stop.apply(lambda x:' '.join(x))#最终结果
```

```
x_cut11.head()
```

最终效果及对比如下所示：

```
0          A市西湖建筑集团占道施工有安全隐患
1     A市在水一方大厦人为烂尾多年，安全隐患严重
2           投诉A市A1区苑物业违规收停车费
3      A1区蔡锷南路A2区华庭楼顶水箱长年不洗
4          A1区A2区华庭自来水好大一股霉味
Name: 留言主题, dtype: object
```

图 4.3　最初数据集

```
0          [A, 市, 西湖, 建筑, 集团, 占, 道, 施工, 有, 安全隐患]
1     [A, 市, 在水一方, 大厦, 人为, 烂尾, 多年, ，, 安全隐患, 严重]
2           [投诉, A, 市, A1, 区苑, 物业, 违规, 收, 停车费]
3      [A1, 区, 蔡锷, 南路, A2, 区华庭, 楼顶, 水箱, 长年, 不洗]
4           [A1, 区, A2, 区华庭, 自来水, 好大, 一股, 霉味]
Name: 留言主题, dtype: object
```

图 4.4 分词后数据

```
0          A 西湖 建筑 集团 占 道 施工 安全隐患
1            A 在水一方 大厦 人为 烂尾 安全隐患
2          投诉 A A1 区苑 物业 违规 收 停车费
3     A1 蔡锷 南路 A2 区华庭 楼顶 水箱 长年 不洗
4          A1 A2 区华庭 自来水 好大 一股 霉味
Name: 留言主题, dtype: object
```

图 4.5　过滤后数据

图 4.6　附件二词云图

图 4.7　附件三词云图

5. 文本处理

5.1 Word2vec 模型

2013 年,Google 开发了一款开源的词向量计算工具-word2vec,引起了工业界和学术界的关注。Word2vec,是一套用于产生词向量的相关模型集。其包含的模型是浅而双层的神经网络,用来训练从而重新建构语言学的词文本。网络的词表现,还需要猜测相邻位置的输入词,在 word2vec 中词袋模型的假设下,词的顺序并没有那么重要。在训练完成后,word2vec 模型可以将每个词映射到一个向量,以用来表示词与词之间的关系,这个向量被称为神经网络的隐藏层。Word2vec 采用了 Hierarchical Softmax 和 Negative Sampling 两种技术来提高训练词向量性能,Word2Vec 是能够把词语转化为多维词向量的模型,根据词语的上下文预测词向量。它的核心架构包括 CBOW 和 Skip-gram。在开始处理文本数据之前,引入模型复杂度,定义如下: O = E * T * Q

其中,E 表示训练的次数,T 表示训练语料中词的个数,Q 根据模型不同有所差距，E 值不是我们分析的对象,T 与训练语料有关,其值越大模型就越准确,Q 将在下面讲述具体模型时讨论。

NNLM 模型是神经网络概率语言模型的基础模型。在 NNLM 模型中,主要影响训练效率的部分是从隐含层到输出层的计算部分,CBOW 和 Skip-gram 模型主要考虑去掉隐含层。大量训练实例证明新训练的词向量的精确度虽然可能不如 NNLM 模型(具有隐含层),但是可以通过增加训练语料的方法来完善这一过程。[1]

Word2vec 包含两种训练模型,分别是 CBOW 和 Skip_gram(输入层、发射层、

输出层),如下图所示:

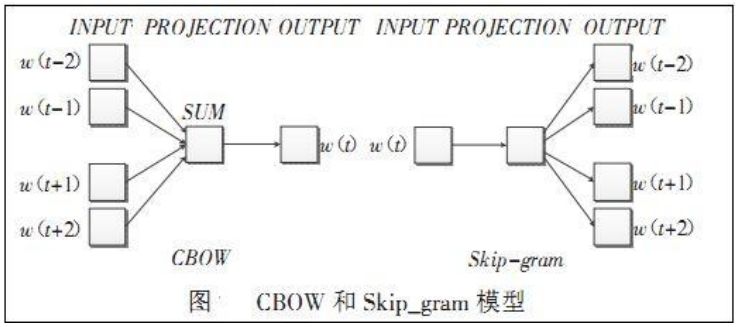

图 5.1 CBOW 和 Skip_gram 模型图

1.CBOW 模型：

该模型可以简单理解为文本上下文的内容决定当前词出现的概率。在 CBOW 模型中,上下文中所有的词对当前词出现概率的影响的权重是一样的,因此被称为 CBOW(continuous bag-of-words model)模型。就好比从一个布袋中取词,只要取出数量足够的词就可以了,词与词之间的顺序变得无关紧要。

2.Skip-gram 模型

Skip-gram 模型是一个简单但却非常实用的模型。

Table 1 The example of Skip-gram

Tri-grams	2-Skip-gram-tri-grams
“中国足球踢得”、“足球踢得真是”、“踢得真是太烂”、“真是太烂了”	“中国足球踢得”、“中国足球真是”、“中国足球太烂”、“中国踢得真是”、“中国踢得太烂”、“中国踢得了”、“中国真是太烂”、“中国真是了”、“足球踢得真是”、“足球踢得太烂”、“足球踢得了”、“足球真是太烂”、“足球真是了”、“足球太烂了”、“踢得真是太烂”、“踢得真是了”、“踢得太烂了”、“真是太烂了”

图 5.2 Skip-gram 模型结果实例图

Skip-gram 模型比较好地解决了页面窗口大小的限制和训练复杂度的问题。根据上表可以知道:在新组成的这 18 个三元词组中,有 8 个词组能够正确反映例句中的真实句意，说明 Skip-gram 模型不仅反映了句子的真实意思，还扩大了语料,三元词组由原来的 4 个扩大到了 18 个。[1]

在本文中，部分问的词向量由基于词袋模型的 Word2vec 确定，而在解决分

类问题时，用 Word2vec 将文本向量化，具体实现及效果如下图所示：

文本向量化

```
def sen2vec(words):
return pd.DataFrame([model.wv[i] for i in words if i in
model.wv]).mean()
train_vec = pd.DataFrame([sen2vec(s) for s in x_train])
```

	0	1	2	3	4	5	6	\
0	-0.000622	0.001402	0.000057	0.000564	-0.000830	0.001821	0.001445	
1	-0.001056	0.001839	-0.000281	0.000594	-0.001001	0.002385	0.002054	
2	-0.000923	0.001786	0.000041	0.000485	-0.000801	0.002286	0.001735	
3	-0.000667	0.001452	0.000030	0.000579	-0.000857	0.001881	0.001515	
4	-0.000470	0.001159	0.000065	0.000512	-0.000716	0.001481	0.001221	
5	-0.000723	0.001757	-0.000077	0.000389	-0.000793	0.002067	0.001688	
6	-0.000769	0.001566	-0.000031	0.000613	-0.000917	0.002020	0.001675	
7	-0.000815	0.001734	0.000100	0.000533	-0.000974	0.002034	0.001733	
8	-0.000807	0.001715	-0.000022	0.000526	-0.000612	0.002051	0.001640	
9	-0.000715	0.001406	-0.000015	0.000628	-0.000853	0.001900	0.001493	
10	-0.000564	0.001313	0.000005	0.000552	-0.000753	0.001729	0.001431	
11	-0.000661	0.001344	-0.000088	0.000636	-0.000917	0.001822	0.001522	

图 5.3　文本向量化结果图

5.2 TF-IDF 算法

TF-IDF(词频-逆文档频率)算法是一种统计方法,用来评估一个字词对一个文件集或一个语料库其中的一份文件的重要程度。字词的重要性与它在文件中出现的次数成正比递增,同时也会和它在语料库中出现的频率成反比递减。这种算法在数据挖掘实现、文本处理和信息检索等方面有着广泛的应用。

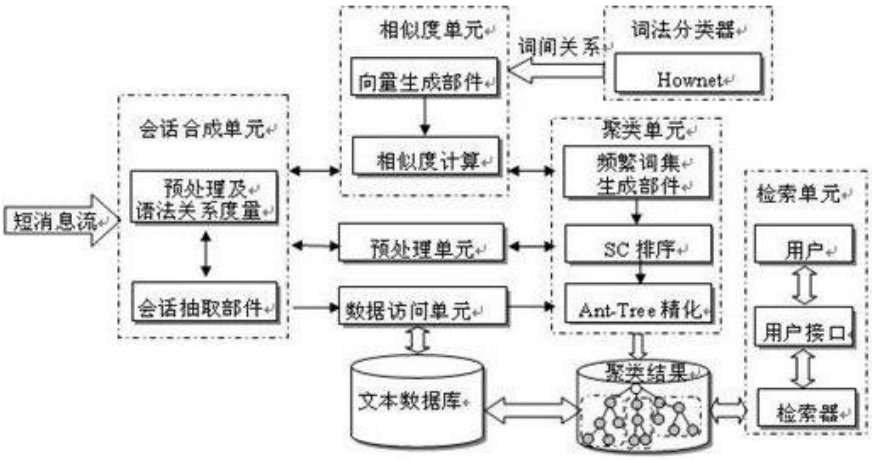

图 5.4　TF-IDF 算法实现流程图

TFIDF 的主要思想如下：若某个词或短语在一分文本文件中出现的频率（TF）较高,且在其他文本文件中很少出现,那么就可以认为这个词或者短语有不错的类别区分能力,适合用作分类指标。TF-IDF 实际上就是 TF*IDF,其中 TF(Term Frequency),是词条在文章 Document 中出现的频率;IDF(Inverse Document Frequency)的主要思想是指：若包含某个词（Word）的文档越少,则这个词的区分度就越大,也就是 IDF 越大。说到怎样提取一份文本资料的关键词,我们可以计算这份文本文件中出现的所有名词的 TF-IDF,TF-IDF 越大,就说明这个名词对这篇文章的区分度就越高,取 TF-IDF 值较大的几个词,就可以当做这篇文章的关键词。

（1） TF 是词频(Term Frequency)

$$TF_w = \frac{\text{在某一类中词条}w\text{出现的次数}}{\text{该类中所有的词条数目}}$$

图 5.5　计算公式 1

TF 是词频(Term Frequency)，代表关键字（词条）在文本中出现的频率。一般情况下会做归一化处理(通常是词频除以文章总词数)，从而防止它偏向长的文件。计算公式为：

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

图 5.6　计算公式 2

其中 ni,j 是这个词在文件 dj 中出现的次数，分母则表示文件 dj 中所有词汇出现次数的总和；

（2） IDF 是逆向文件频率(Inverse Document Frequency)

$$\text{逆文档频率} = \log \frac{\text{语料库的文档总数}}{\text{包含该词的文档数}+1}$$

图 5.7　计算公式 3

逆向文件频率 (IDF)：某一个指定词语的 IDF 可以用总文件数量除以包含该词的文件数目，再把得到的结果取对数得到最终结果。假如包含词条 t 的文档越少, IDF 越大，就说明该词条的类别区分能力比较优秀。公式：

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

图 5.8　计算公式 4

其中，|D|　是语料库中包含的的总文件数。|{j:ti∈dj}|表示包含词语 ti 的文件数目（即　ni,j≠0　的文件数目）。若该词语不在语料库中，则分母为零，因此一般情况下使用　1+|{j:ti∈dj}|

即：$IDF = log(\frac{\text{语料库的文档总数}}{\text{包含词条}w\text{的文档数}+1})$, 分母之所以要加1，是为了避免分母为0

图 5.9　计算公式 5

（3） TF-IDF 实际上是：TF * IDF

词频－逆文档频率＝词频＊逆文档频率

某个指定件内的高词语频率，还有该词语在整个文件集合中的低文件频率，都可以产生出有高权重的 TF-IDF。因此，TF-IDF 倾向于保留比较重要的词语，而把一些常见的词语过滤掉。公式：TF－IDF＝TF＊IDF

6. 模型选择

6.1 Logistics 模型

Logistic 回归是一种广义线性回归（generalized linear model），因此与多重线性回归分析有很多相同之处。它们的模型形式基本上相同，都具有 w‘x+b，其中 w 和 b 是待求参数，其区别在于他们的因变量不同，多重线性回归直接将 w‘x+b 作为因变量，即 y =w‘x+b，而 logistic 回归则通过函数 L 将 w‘x+b 对应一个隐状态 p，p =L(w‘x+b),然后根据 p 与 1-p 的大小决定因变量的值。如果 L 是 logistic 函数，就是 logistic 回归，如果 L 是多项式函数就是多项式回归。[2]

Logistic 回归和线性回归最大的区别就在于 Y 的数据类型。线性回归分析的因变量 Y 是属于定量数据的，而 Logistic 回归分析中的因变量 Y 是属于分类数据。

Logistic 回归分析主要是在研究影响关系的方面，即 X 对于 Y 的影响情况。其中，Y 是定类数据，X 既可以是定量数据也可以是定类数据。用途主要在预测和判别，倘若已经建立了 logistic 回归模型，则可以根据模型，预测在不同的自变量的情况下，发生某病或某种情况的概率有多大。

logistic 回归的因变量可以是二分非线性差分方程类的，也可以是多分类的，但是二分类的更为常用，也更加容易解释。所以实际中最为常用的就是二分类的 logistic 回归。

6.2 SVM 模型

支持向量机(support vector machines, SVM)是一个二分类模型,它的基本模型是定义在特征空间上间隔最大的线性分类器,最大间隔使它有别于感知机;SVM 还包括了核技巧,这使它实质上成为了非线性分类器。SVM 的学习策略就是间隔最大化,可以将其形式化为一个求解凸二次规划的问题,即等价于正则化的合页损失函数的最小化问题。SVM 的学习算法就是最优化求解凸二次规划的算法。它有一个重要性质:当训练完成后,大部分的训练样本都不需要保留,最终模型仅与支持向量有关。支持向量机的目的在于求得最优的几何间隔最大的超平面,当样本数据是线性可分时,这里的间隔最大化又被称为硬间隔最大化(若训练数据近似可分，则称为软间隔)支持向量机的学习算法可以表示为下面的约束最优化问题:maxw, b ϒ =w·xi+b||w||； s.t. yi(w·x+b)≥1, i=1, 2, 3, ⋯, N

6.3 k-means 模型

1.K-means 算法，也称为 K-平均或者 K-均值，是一种聚类分析算法，用迭代求解的方式对数据进行聚类处理。一般作为掌握聚类算法的第一个算法。

2.在具体操作时，会把数据预分为 K 组，而后随机地选取 k 个对象作为一开始的聚类中心，再计算每个元素到聚类中心的距离，依据“就近原则”给每个聚类中心分配元素对象。

3. 当对象全部被分配时，会再次重复计算聚类中心，程序重复执行这个过程直到满足没有（或最小数目）对象被重新分配；没有（或最小数目）聚类中心再次发生变化；误差平方和局部最小三个条件之一时终止计算。

4.在对样本进行聚集的过程往往是以样本之间的距离作为指标来划分。

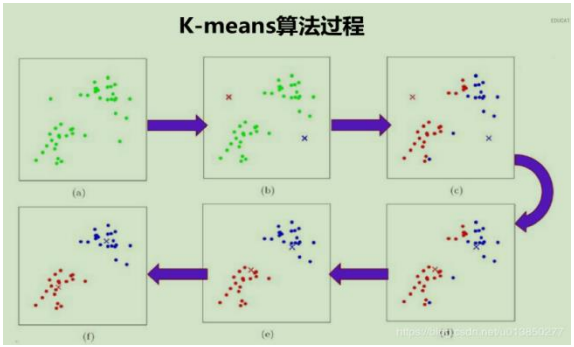

图 6.1 K-means 算法过程

6.4 模型确立

下图为第一问同一初步数据处理下用 tf-idf 计算空间向量时不同模型的预测结果：

Logistics 模型：

	precision	recall	f1-score	support
交通运输	0.97	0.66	0.78	155
劳动 和 社会保障	0.81	0.90	0.86	472
卫生 计生	0.90	0.70	0.79	216
商贸 旅游	0.90	0.70	0.79	293
城乡建设	0.66	0.93	0.77	500
教育 文体	0.92	0.84	0.88	439
环境保护	0.96	0.75	0.85	228
avg / total	0.85	0.82	0.82	2303

图 6.2　Logistics 模型结果

SVC 模型：

	precision	recall	f1-score	support
交通运输	0.00	0.00	0.00	155
劳动 和 社会保障	0.00	0.00	0.00	472
卫生 计生	0.00	0.00	0.00	216
商贸 旅游	0.00	0.00	0.00	293
城乡建设	0.22	1.00	0.36	500
教育 文体	0.00	0.00	0.00	439
环境保护	0.00	0.00	0.00	228
avg / total	0.05	0.22	0.08	2303

图 6.3　SVC 模型结果

综上所述以及结合数据最终准确率不难发现 Logistics 模型性能远优于 svm 模型，故我们组第一问决定选用 TF-IDF 结合 Logistics 模型对附件 2 进行文本分类。

7. 评估标准

7.1 f-score

f-measure 是一种统计量，F-Measure 又称为 F-Score，F-Measure 是 Precision 和 Recall 加权调和平均，是 IR（信息检索）领域的常用的一个评价标准，常用于评价分类模型的好坏。在 f-measure 函数中，当参数 α=1 时，F1 综合了 P 和 R 的结果，当 F1 较高时则能说明试验方法比较有效。[4]

很多时候我们需要综合权衡 Precision 和 Recall 这两个指标，这就引出了一个新的指标 F-score。这是综合考虑 Precision 和 Recall 的调和值

$$F-Score=(1+\beta^2)\cdot\frac{Precision\cdot Recall}{\beta^2\cdot Precision+Recall}$$

图 7.1　F-Score 计算公式

当 β=1 时，称为 F1-score，这时，精确率和召回率都很重要，权重相同。当有些情况下，我们认为精确率更重要些，那就调整 β 的值小于 1，如果我们认为召回率更重要些，那就调整 β 的值大于 1。

7.2 热度评估指标

在第二问中关于热度评价，我们小组决定采用（点赞数-反对数）/文本数量的值作为热度评估指标，类似于求出同类文本的平均值，其中点赞数视为正数，反对数视为负数，同类文本的个数视为数值数量。

7.3 第二问相关要求指标

同样是第二问中，文本所给的表头中的问题描述、时间范围以及地点人群决定分别用 LDI 模型、pyhanlp 以及 datetime 时间库进行拟合。其中，时间范围是固定的，只需求出最大最小值。如下图所示：

```
第四类所有文本的时间： [Timestamp('2017-06-08 17:31:20'), Timestamp('2019-04-28 17:32:51'), Timestamp('2018-05-1
7 08:32:04'), Timestamp('2019-11-05 10:31:38')]
时间范围： 2017-06-08 2019-11-05
```

图 7.2　时间范围结果

而另外二者的识别则会产生一定的误差，在此我们决定先将其识别出来以后

再进行第二步的误差缩小，判断二者的识别的正误与否。

其中，LDI 模型同 k-means 聚类一样，生成的类别数影响较大，在此我们决定将其和上述聚类类别数量达成一致，LDI 模型的实现如图所示：

```
yy = text
dictionary = corpora.Dictionary(yy)
corpus = [dictionary.doc2bow(text) for text in yy]
corpora.MmCorpus.serialize('corpus.txt',corpus)
tfidf_model = models.TfidfModel(corpus)
corpus_tfidf = tfidf_model[corpus]
lda = models.LdaModel(corpus_tfidf,id2word=dictionary, num_topics=10)
corpus_lda = lda[corpus_tfidf]
lsi = models.LsiModel(corpus_tfidf,id2word=dictionary, num_topics=20)
corpus_lsi = lsi[corpus_tfidf]

#生成5个主题描述
print(lsi.show_topics(num_topics=4, num_words=5))
```

```
[(0, '0.279*"劳动" + 0.279*"东路" + 0.277*"A5" + 0.277*"区" + 0.262*"小区"'), (1, '-0.4
58*"学院" + -0.404*"强制" + -0.404*"实习" + -0.358*"学生" + -0.355*"经济"'), (2, '-0.41
7*"的" + 0.254*"扰民" + -0.232*"污染" + -0.232*"附近" + -0.232*"空气"'), (3, '0.343
*"？" + 0.334*"什么" + 0.334*"时候" + 0.334*"能" + 0.210*"假"')]
```

图 7.3　LDI 模型的实现

而相关地点或人群的识别则用 pyhanlp 库，类似于 jieba 库的使用，实现效果如下图所示：

```
from pyhanlp import *
segment = HanLP.newSegment().enableOrganizationRecognize(True)
for i in suoyin(data,4,2):
    print(HanLP.segment(i))
```

```
[A/nx, 市/n, 经济学院/nit, 体育学院/nit, 变相/vd, 强制/vd, 实习/vn]
[A/nx, 市/n, 经济学院/nit, 强制/vd, 学生/nnt, 实习/vn]
[A/nx, 市/n, 经济学院/nit, 强制/vd, 学生/nnt, 外出/vi, 实习/vn]
[A/nx, 市/n, 经济学院/nit, 组织/n, 学生/nnt, 外出/vi, 打工/vi, 合理/a, 吗/y, ？/w]
```

图 7.4　相关地点或人群的识别

7.4 自定义加权评估标准

利用文本相似度、回复时长和回复字数，三者的结果人为进行恰当的加权后返回一个具体数值，把这个数值作为最后的评估指标。

7.4.1 文本相似度

文本相似度，顾名思义就是衡量两篇文本的相似程度. 文本作为承载语义信息的一种重要方式，传统的文本表示采用向量空间模型来表达语义信息，这种方式未考虑到特征词的顺序以及上下文语义理解，造成高维稀疏以及计算效率低的

问题[3]

1. 词向量平均

简单的对句子中的所有词向量取平均，十分的简单方便易于操作，但是因为没有考虑到单词和单词之间的相关顺序以及忽略了词和词之间的相关意思，无法更好的通过词义分析句子与句子之间的关系。

2. tfidf 加权

指对句子中的所有词向量根据 tfidf 权重加权求和，相较于上面的词向量均而言，效果更好，因为它考虑到了 tf-idf 权重，所以可以凸显出权重大的词也就是所谓的关键词。不过它和上述方法一样没有顾及到词与词之间的语序。

3.余弦计算

顾名思义，就是将文本变成向量之后计算他们在空间中的余弦大小，然后以此作为衡量俩文本之间的差异的一个指标。两个向量相似度越大，距离越小；反之相似度越小，距离越大。在计算文本相似度时，首先要让文本转化为可以用于计算的数值，经过分词处理后，通过计算 TF-IDF 然后生成向量，经过词频统计和加权处理后进行余弦相似性的计算。计算公式：

$$\mathrm{sim}(A,B)=\cos(\theta)=\frac{A\cdot B}{\|A\|\|B\|}=\frac{\sum_{k=1}^{n}A_k\times B_k}{\sqrt{\sum_{k=1}^{n}A_i^2}\times\sqrt{\sum_{k=1}^{n}B_i^2}}$$

图 7.5　余弦计算公式

在本文计算文本相似度的过程中：

（1）利用 gensim 库将所有的留言详情作为语料库，加载到字典去，并且结合 tf-idf，然后将需要评价的答复意见再单一抽取出来进行对比他们二者之间的相似度。如下图所示：

抽取的第一条答复意见和不同的留言详情之间的相似度：

```
texts = []
for each in text1:
    texts.append(list(each))
from gensim.models.word2vec import Word2Vec
```

```
from gensim import corpora, models, similarities
model = Word2Vec(size=400, min_count=5)
dictionary = corpora.Dictionary(texts)
num_features = len(dictionary.token2id.keys())
corpus = [dictionary.doc2bow(text) for text in texts]
tfidf = models.TfidfModel(corpus)
liuyan, dafu = data.loc[0,["留言详情","答复意见"]]
text2 = stop([],jieba.lcut(dafu))
new_vec = dictionary.doc2bow(text2)
sim = index[tfidf[new_vec]]for i in range(0,5):
print('第', i+1, '句话的相似度为：', sim[i])
```

```
第 1 句话的相似度为： 0.06460352
第 2 句话的相似度为： 0.006930472
第 3 句话的相似度为： 0.0
第 4 句话的相似度为： 0.026670821
第 5 句话的相似度为： 0.0
```

图 7.6　相似度结果

抽取的前 20 条答复意见和各自相对应答复意见之间两两的文本相似度：

```
sim_1 = sim[0]
for i in range(0,20):
    text2 = stop([],jieba.lcut(data.loc[i,"答复意见"]))
    new_vec = dictionary.doc2bow(text2)
    index = similarities.SparseMatrixSimilarity(tfidf[corpus], num_features)
    sim = index[tfidf[new_vec]]
    print(sim[i])liuyan, dafu = data.loc[0,["留言详情","答复意见"]]
text2 = stop([],jieba.lcut(dafu))
new_vec = dictionary.doc2bow(text2)
sim = index[tfidf[new_vec]]for i in range(0,5):
        print('第', i+1, '句话的相似度为：', sim[i])
```

0.06460352
0.06329634
0.036884252
0.046766475
0.02047522
0.0
0.08689733
0.016076688
0.086027235
0.13490413
0.025742754
0.054464832
0.084324405
0.023678344
0.31786096
0.14529532
0.015404403
0.47440273
0.0013299509
0.46247935

图 7.7　前二十条相似度

通过和附件 4 之间的比对可以发现比对结果不尽人意，文本间的相似度均值统一是偏小的，给后期计算容易带来偏差，除此之外还有，比如第六条的文本相似度为 0，但如图示的答复详情可以发现其答复意见十分中肯。印证上述内容偏差较大。

网友“A00077538”：您好！针对您反映A3区含浦镇马路卫生很差的问题,A3区学士街道、含浦街道高度重视，现回复如下：您留言中反映的含浦镇在2013年已经析出两个街道，分别是学士街道和含浦街道，鉴于您问题中没有说明卫生较差的具体路段，也没有相应的参照物，同时您也未留下联系方式，请您看到回复后，致电学士街道0731-0000-00000000或者含浦街道0731-0000-00000000反映相关问题。感谢您对我们工作的关心、监督与支持。2019年4月24日

图 7.8　中肯答复实例

（2）利用 Word2vec 利用词向量均值化表示成相对应的句向量，然后计算两个句向量之间的余弦大小。直接比较同一留言情况和答复意见二者的相似度。

```
def cos_dist(vec1,vec2):
        dist1=float(np.dot(vec1,vec2)/(np.linalg.norm(vec1)*np.linalg.norm(vec2)))
         return dist1
vecnew = pd.DataFrame([model.wv[i] for i in x1 if i in model.wv]).mean()
vec2 = pd.DataFrame([model.wv[i] for i in x2 if i in model.wv]).mean()
print(cos_dist(vecnew,vec2))
```

0.04744457866855103

图 7.9　所得值

虽然求得值也偏小，但两两之间的比对可以有效避免过大语料库中一些冗杂词汇对相似度的影响，使最后的比对效果误差过大。

综上所述，我们组决定利用 Word2vec 向量化之后直接求两者余弦得出他们的文本相似度。

7.4.2 回复时长

通过代码得出回复时长来确定时效性，本文直接利用 datetime 库提取出附件中的时间变量，变为标准的日期时间之后再进行加减操作，得出回复的时长，然后赋予相合适的权重。

7.4.3 回复字数

通过对评论字数爬取来展现评论的有效性以及真实性。数字越多不一定回复质量越高，但是根据附件内容不难发现，字数太少则一定不会是太合乎标准的，如所示留言，字数相比过少，质量过低

网友：您好！留言已收悉
网友：您好！留言已收悉 。

图 7.10　字数过少评论实例

所以我们有理由将回复字数的多少也作为回复质量的评价指标之一。

7.4.4 数据标准化

1.z-score 标准化(zero-mean normalization)

也叫标准差标准化。将数据按比例缩放，让它落到一个相较而言比较小的特定区间，标准化后的数据科能有正也有负数值，不过其绝对值一般不会过大。

计算时要先减去均值，再除以均方根。这种方法在提高数据可比性的同时，也降低了数据的解释性，是使用的最多的数据标准化方法。结果输出时：每个属

性值的均值为 0，方差为 1，图形呈正态分布。

公式：x* = (x- μ)/ σ

（其中 μ 表示所有样本数据的均值，σ 表示所有样本数据的标准差。）

2. 正则化（normalization）

正则化的过程是把每个样本都缩放到单位范数（即使每个样本的范数都是一），在使用二次型（点积）或是其他一些核方法计算样本之间的相似性时，这种操作十分有用。Normalization 的主要思想是分别计算每一个样本的 p-范数，然后用样本中的每一个元素除以得到的范数，在这样的处理后，每个样本的 p-范数（l1-norm, l2-norm）都等于 1。

如图所示为第一条答复和第一条留言的上述三个判断指标列表，对其分别正则化和标准化：

正则化和标准化

```
all_p = [cos_dist(vecnew,vec2),(y-x).days,len(text2)]
print("初始值：",all_p,"  依次为余弦值，回复时长（天数），字数")
print("标准化：",preprocessing.scale(all_p))
print("正则化：",normalization(all_p))
```

```
all_p = [cos_dist(vecnew,vec2),(y-x).days,len(text2)]
print("初始值：",all_p,"  依次为余弦值，回复时长（天数），字数")
print("标准化：",preprocessing.scale(all_p))
print("正则化：",normalization(all_p))
```

```
初始值： [0.04744457866855103, 15, 161]   依次为余弦值，回复时长（天数），字数
标准化： [-0.80757309 -0.6016334   1.4092065 ]
正则化： [0.          0.09290039 1.        ]
```

```
def jiaquan(x):
    x = preprocessing.scale(x)
    return(3*x[0] + (-2)*x[1] + 5*x[2])
jiaquan(all_p)
```

```
5.826580001344816
```

图 7.11　第三问加权效果

不难发现因为文本相似度也就是余弦大小，同文本字数之间差异过大，正则化固定范围在（0,1）之后反而表述不出正确的数据大小，可能会造成数据偏差，所以本文决定对数据进行标准化，然后进行加权处理。

针对加权处理，因为余弦值和字数都是越大越好，而回复时长则是越小越好，所以对回复时长的权重做了负值处理，以此表示其与另外两个的逆向关系。此外，权重采用最传统的比例，即 3:2:7（其中 2 为负值权重）的形式计算最终答复质量的评估效果。

8、总结

本文的主要目的是利用目前已经掌握的知识，通过大量数据留言内容去发掘出群众的民生焦点，挖掘出隐藏在留言内容之中的民生关注点，为后续政府利用前面的海量数据更好的去有针对性的实施措施，生产出一系列利民、惠民措施。

首先是要将大量留言数据有序的归类统一起来，由于附件直接给了留言主题所以我们为了方便也防止数据特征过多过大造成维度灾难直接提取了留言主题作为特征，本文最明显的文本处理特点就是中文不同于英文的复杂性，所以在数据预处理时要比英文预处理多一个分词的步骤，这一步可以自行用算法又或者是用 python 自带的中文库实现。除此之外，为了更好地提取有利特征，我们需要将分词之后的文本过滤掉一些用处甚微的词语和符号，以防对后续模型拟合时造成误差影响。其次是第二问中的无标签分类，其中所有数据的预处理就和我上面画的流程图一样是一样的操作，只是不同的要求可能会让后续操作不太相同。其中热度指标则是采用比较常见比较容易想到的点赞数减去反对数的平均。从这里开始后续操作结果就和操作者有着比较密切的联系。关于答复详情的评价系统，也是和上述一样采取较为容易的思路去评价，然后进行加权，而其中如何加权，怎么样的权重才是最合适的又是一个影响较大比较重要的值。

从结果来看，我们的预测效果并没有想象中的那么好，可能有些地方还可以做一些改善、增强从而提高准确率，又或是可能另一种方法比此方法更好。比如最后关于答复的针对性评价，可能会因为人为的赋权导致权重并不准确从而使结果也不精确。总而言之，这次实验中还有许多值得改进完善的地方，这也是我们下一次需要去注意并加强的。

参考文献：

[1] 熊富林,邓怡豪,唐晓晟.Word2vec 的核心架构及其应用[J].南京师范大学学报(工程技术版),2015,15(01):43-48.015,25(02):145-148.

[2] 谢花林, 李波. 基于 logistic 回归模型的农牧交错区土地利用变化驱动力分析——以内蒙古翁牛特旗为例[J]. 地理研究, 2008, 027(002):294-304.

[3] 徐鑫鑫,刘彦隆,宋明.利用加权词句向量的文本相似度计算方法[J].小型微型计算机系统,2019,40(10):2072-2076.

[4] 百度百科“f-measure”词条[EB]/[OL].词条网站

[5] 斯坦福大学自然语言处理实验室网站. Evaluate of clustering