基于文本挖掘的留言整理及答复评价

摘 要

网络问政是一种新型的民主形式，既提供给政府更大的范围以收集民意，也提供给公民一种介入政治生活、参与政治过程、行使公民权利的新路径。然而，各类社情民意相关的文本数量不断攀升，如此庞大的文本数据仅依靠人工来划分和整理显然是非常困难的。如何在降低人力物力的要求的同时，更方便快捷地处理这些文本数据，建立健全相应的问政机制，使其良性发展，是当下急需解决的重要课题之一。

本文首先对整体数据进行预处理，去除噪声。如：格式转换、去掉符号、整体规范化等，再遍历读取一个文件下的每个文本，然后进行简单分类处理，使后续建模使用的初步文件结构更清晰，操作更方便。

针对问题一，本文建立了关于附件 2 的文本分类模型并进行模型评价。首先，读取经过预处理后的附件 2 的文件，按照 80%:20%的比例，随机拆分得到训练集和测试集。为了测试代码的准确性，我们特意将一些文档放在错的类别的语料库中，然后用 jieba 的精准模式进行分词，再对处理之后的训练集和测试集文本用 TF-IDF 算法进行单词权值的计算，构建训练集向量空间，统计词频，生成每个文本的词向量，并去掉停用词，最后运用朴素贝叶斯分类算法，得到较为准确的分类结果(见图 3-7)。分析结果表明，我们提出的模型和方法能正确的检测出大部分文档类型。最后，我们使用 F-Score 得分对此分类方法进行评价，得出的评价指数为 0.8977(详见表 3-1)。

针对问题二，本文建立了关于附件 3 的热点问题的文本挖掘模型。首先，我们将附件 3 的主题整理为文本集，并进行分词和关键词提取，然后进行词频统计并排序，再利用 TF-IDF 信息抽取为主题词表。根据 word2vec 模型进行相似度统计，汇成分类表格提取热度主题，进行热度指数评价分析，得到前五个热门主题，最后汇总生成文件“热点问题表”和“热点问题留言明细表”。“热点问题表”中第一的热点主题是“A 市 A7 县居民反映交通和施工扰民等问题”，“热点问题留言明细表”中共计有 1473 条留言。

针对问题三，建立关于附件 4 的答复意见的质量评价模型。评价方案同时从三个角度出发：答复的相关性、完整性、可解释性。答复的相关性和完整性都是是用文本挖掘模型找到关键词进行关联度分析，得到相关性和完整性的评价指数。答复的相关性和完整性是从留言答复平台内部进行评价的，而可解释性是从网友角度进行评价的。可解释性又可分为说服性和有效性，综合二者即可获得网友反馈评分后的可解释性的评价指数。最后综合相关性、完整性、可解释性的评价指数得到对答复意见的综合评价。

关键词：朴素贝叶斯分类模型；文本分类；主题热度计算模型；主题聚类算法

目 录

一、引言

1.1 问题背景

随着互联网的日益普及和上网人数的逐渐增加，我国公民已从最初的通过网络了解时事新闻、发表自己的见解，逐渐发展为如今的运用网络参政问政、表达诉求。据统计，2008 年有 11 家媒体开办网络问政平台，2009 年达到 21 家。截至 2010 年 6 月 30 日，全国 31 个省、自治区直辖市中，有 23 个地方、43 个媒体开办了省级网络问政平台(不包括人民网)，已经开办的省级网络问政平台达到了 74.2%。中部地区和东部沿海省份基本上都开通了网络问政平台[1]。

然而，此种新型问政方式也面临着不少困难。数据显示，仅 2018 年，广大网友就在江西的大型网络问政平台“问政江西”中发帖超 12 万条[2],日均 300 条，也就是说大概每四分钟便有一名网友发来求助、投诉，或者建议贴文。面对如此庞大的各类社情民意相关的文本数据量，我们急需找到便捷高效的方法来处理这些数据,这正是本文将要探讨并解决的问题。本文主要通过建模研究了以下问题：

(1)根据附件 1 中的三级标签体系，建立关于附件 2 中留言内容的文本分类模型，并用 F-score 对分类结果进行评价。

(2)根据附件 3 将某一时段内反映特定地点或特定人群问题的留言进行归类，定义合理的热度评价指标，并给出评价结果。

(3)根据附件 4 相关部门对留言的答复意见，从答复的相关性、完整性、可解释性等角度对答复意见的质量给出一套评价方案，并尝试实现。

1.2 问题分析

本文要解决的是网络问政平台处理流程的优化问题。首先，我们提出网络问政平台的一般处理流程，如图 1-1 所示。

问题一和问题二要求从网络问政平台的角度，处理平台中大量信件的留言划分和热点整理问题，以便解决人工处理时出现的工作量大、效率低、出错率高等问题。问题三是从留言答复方面，建立对留言答复进行统一标准的评价系统，用来提升政府的管理水平和施政效率。

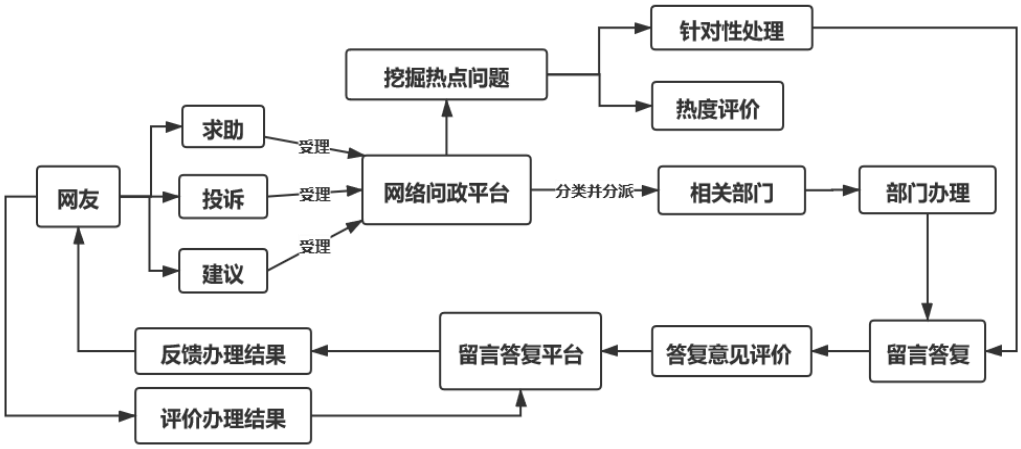


图 1-1 网络问政平台处理流程

1.2.1 对问题一的分析

由图 1-1 可知，面对大量来自网友们的留言内容，工作人员在处理时会先按照一定的划分标准，比如附件 1 中的一级标签，将留言粗略地进行分类。但是如果都由工作人员根据经验来完成分类，则存在工作量大、效率低、且差错率高等问题，而且会消耗大量的人力及物力，造成一定程度的资源浪费。

问题一是给群众留言进行分类，分类的目的是对新的群众留言贴上合适的标签。由于一个留言可能会与多个类别相关，所以，问题一属于多标签分类问题。要实现留言的自动分类，我们首先要对训练集中的留言文本进行分析，设计一个合理的分类模型。本文主要采用贝叶斯分类算法来解决此问题。

文本分类的关键在于准确提炼留言中的中心思想，而提炼中心思想的方法则是抽取文本或句子中的关键词，并以此作为特征，最后基于这些特征去训练分类器，达到正确分类的目的。

1.2.2 对问题二的分析

由图 1-1 可知，网络问政平台在分类并分派给相关部门时，同时也在挖掘热点问题，并且计算热点指数，形成热点问题表。

问题二就是希望寻找热点问题。本文的热点主题抽取主要分为三个阶段。一是实验数据采集阶段，主要以附件 3 中给出数据为实验文本集；二是信息抽取阶段，主要通过中文分词工具进行信息抽取，再进行词频统计并排序；三是主题词形成阶段，利用 TF-IDF 信息确定主题词表，然后根据相似度分析形成热度主题，并进行模型评价。

当前社会每天都有着不同的热点问题。比如微博的“热搜”，微信指数 WCI 等。如何能反映出某一问题的热度，一般是从阅读量、点赞数、回复数、提及数等方面来综合考虑。本文在附件 3 的基础上，建立了通过热点问题的数量、点赞数、反对数等来综合计算热度的评价模型。

1.2.3 对问题三的分析

由图 1-1 可知，留言经过相关部门处理后，会以留言答复平台把答复信提交给相应网民，同时获得一个反馈评价。再此之前，也可以根据相应方案提前对答复内容的质量给出评价。问题三的就是希望建立一个从答复的相关性、完整性、可解释性等角度对答复质量进行合理评价的方案。这里的“相关性”、“完整性”和“可解释性“的说明如下：

(1)“相关性”指两个变量的关联程度。一般地，从散点图上可以观察到两个变量有以下三种关系之一：两变量正相关、负相关、不相关。如果一个变量高的值对应于另一个变量高的值，相似地，低的值对应低的值，那么这两个变量正相关。反之，如果一个变量高的值对应于另一个变量低的值，那么这两个变量负相关。如果两个变量间没有关系，即一个变量的变化对另一变量没有明显影响，那么这两个变量不相关[3]。通过关键词抽取方法，获得留言时间和答复意见的关键词后，可计算文本相关度。

(2)“完整性”在不同的领域有着不同的含义。本文指“信息的完整性”，其具体含义是指包括所有与信息使用者要做的事情相关的信息。如，在一个风险投资的计划书中，如果没有主要原材料的成本分析，则信息完整性就会大打折扣。信息的完整性是与接收信息者的目的密切相关的[4]。所以留言答复的完整性则是看留言详情里的所有问题是否在留言答复中得到解答，由此我们判断答复的完整性质量。

(3)“可解释性”(如图 1-2 所示)不同于前述两个角度，它是基于网友方面对答复的反馈来进行评价。网友的反馈主要分为两个方面，一方面是说服性，指网友收到留言答复后觉得该答复是否有说服力，即是否对留言详情中的所有问题都做出了解答；另一方面是有效性，指网友觉得答复里的措施是否有效地解决了问题。

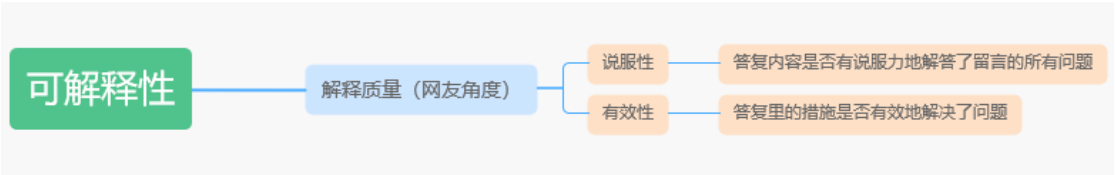


图 1-2　可解释性的解释

二、数据预处理

文本预处理即实验前的文本处理过程。由于数据的不完整性，以及质量好坏不一等因素导致我们无法直接对现有数据进行挖掘。为了提高数据挖掘的质量，在数据挖掘之前需进行数据预处理，其主要任务是去除噪声，如格式转换，去掉符号，整体规范化等等都属于数据预处理。

2.1 问题一数据预处理

(1)首先对附件二中的居民留言文本数据，按照附件一的一级标签转换成 7 个以一级标签名命名的文件夹(如图 2-1 所示)，然后再利用 Excel 中的 VBA 将每一行导出为一个单独的 txt 文本，使得每个 txt 文本的文件名为该行留言的留言编号后加 1 的数字名，并存于文件夹中(如图 2-2 所示)。最后按照 80%:20%的比例，随机地把全体数据分成训练(data)集和测试(test)集(如图 2-3 所示).

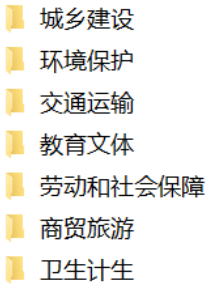


图 2-1 文件夹命名

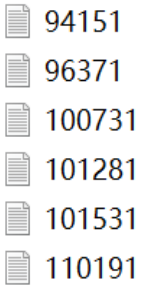


图 2-2 TXT 命名

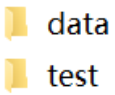


图 2-3 训练集和测试集

(2)得到(1)中数据后，还要再除去各种符号，制表符，空格，回车等不必要的干扰因素。

如：除去换行符：

```
result = (str(content)).replace("\r\n", "").strip()
```

2.2 问题二数据预处理

2.2.1 文本预处理

(1)读取居民留言文本数据。删除标点符号、数字及其它特殊字符。

(2)因为计算机不能处理这么高度抽象的文字，所以我们需要使用问题一所用的 jieba 进行分词，然后用空格将词分开，形成类似英文那样的文本，方便计算机处理。

(3)删除停用词(“的”“地”“我们”等)。

(4)进行词频数计算。代码如附录 2 所示，结果如图 2-4 所示。

图 2-4 词频统计结果

2.2.2 Word2vec 的预处理：

首先把所有需要进行训练的词汇编上序号，比如 1-5000 等。随机初始化一个维度为 5000x50 的矩阵，作为待训练的嵌入矩阵。每次取出一个中心词和它的其中一个环境词(环境词是中心词前后的若干 *n* 个词，*n* 越大效果越好，但速度越慢)。

再以环境词编号作行数，从词向量矩阵里取出这一行数据(50 维向量)。将这个 50 维向量作为逻辑回归网络的输入，训练目标是中心词编号相应 One-Hot 向量。在训练的反向传播时计算，不但更新逻辑回归网络的权重矩阵，还要往前多传递一级，把取出的 50 维向量的值也根据目标梯度进行更新。

最后将更新过的 50 维向量重新更新到嵌入矩阵相应的行。重复以上过程，直到所有的中心词都已经被遍历一遍，此时嵌入矩阵值的计算就完成了。

三、问题一的模型建立与求解

3.1 朴素贝叶斯分类器简介

朴素贝叶斯是一种简单但极为强大的预测建模算法。通过查找文献[5]，我们发现相比于其他精心设计的更复杂的分类算法，比如随机森林、决策树、支持向量机等，朴素贝叶斯分类算法是学习效率和分类效果较好的分类器之一。之所以称为朴素贝叶斯，是因为它假设每个输入变量是独立的。这是一个强硬的假设，实际情况并不一定，但是这项技术对于绝大部分的复杂问题仍然非常有效。朴素贝叶斯分类模型是由两种类型的概率组成：每个类别的概率和每个属性的条件概率。朴素贝叶斯分类模型是在贝叶斯原理的基础上设计出来的，利用了贝叶斯中求解“逆向概率”的方法和原理，对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个概率最大，就认为此待分类项属于哪个类别。

朴素贝叶斯文本分类是首先获取留言文本数据，然后进行数据预处理，为了使计算机更好识别需分词、建立词频矩阵和特征词库，最后将数据带入贝叶斯分类模型进行训练。模型训练好后，我们还需要测试，测试只需要将测试数据集带入贝叶斯分类模型中运行即可。朴素贝叶斯进行留言文本分类的流程图如图 3-1 所示：

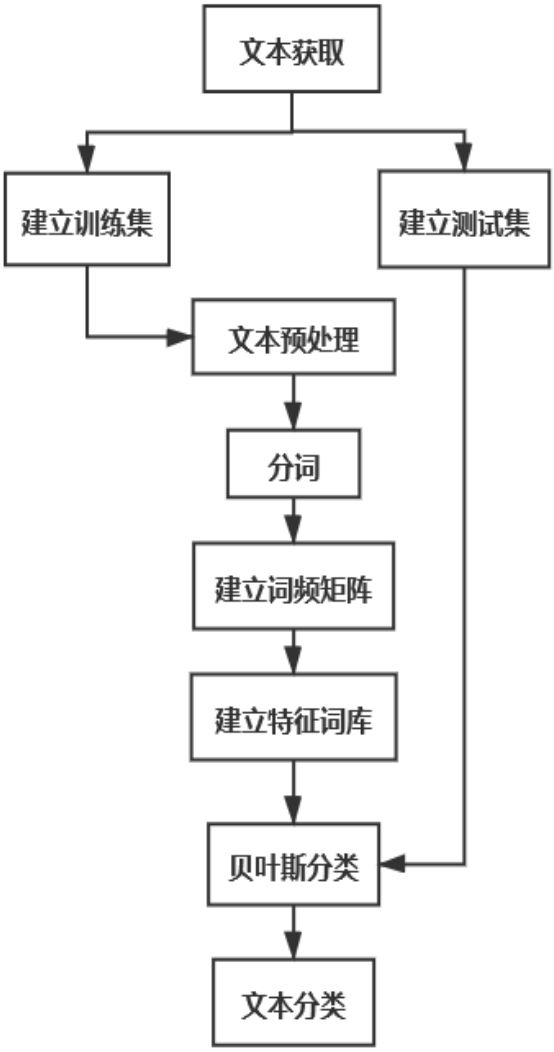


图 3-1 朴素贝叶斯文本分类流程

3.2 问题一的求解

3.2.1 语料选择

通过数据预处理我们得到了训练集和测试集，再建立一个停用词集(图 3-2)。停用词指一些没有起决定性作用的，对文档分类没有帮助，但是在文本中出现次数又很多的词。在文本分析中，为节省存储空间和提高效率，在处理文本之前需要过滤掉这些字和词。我们把停用词集与之前的训练集和测试集放在一起，得到了三个基本文档(图 3-3).

阿
哎
哎
哎
唉
俺
俺
按
按
吧
吧
把
罢
被
呀
哟
们
照
哒
了

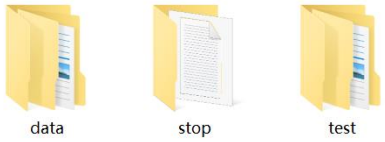


图 3-2 停用词集

图 3-3 基本文档

3.2.2 中文分词

(1)中文分词(Chinese Word Segmentation) 指的是将一个汉字序列切分成一个一个单独的词。分词就是将连续的字序列按照一定的规范重新组合成词序列的过程。中文分词就是将一句话拆分为各个词语，因为中文分词在不同的语境中歧义较大，所以分词极其重要。最常见的分词算法可以分为三大类：基于字符串匹配的分词方法、基于理解的分词方法、基于统计的分词方法。

▸原型：我今天中午吃的小面。

▸分词：我、今天、中午、吃、的、小面。

▸其中　我、的、两个分词属于停用词。

(2) jieba 分词简介：

jieba 分词算法使用了基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能生成词情况所构成的有向无环图(DAG), 再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，对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法

结巴分词的三种模式：

精准模式：试图将句子最精确地切开，适合文本分析。

全模式：吧句子中所有的可以成词的词语都扫描出来，但不能解决歧义。

搜索引擎模式：在精准模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

#精确模式: 我/ 去过/ 清华大学/ 和/ 北京大学/。

#全模式: 我/ 去过/ 清华/ 清华大学/ 华大/ 大学/ 和/ 北京/ 北京大学/ 大学/。

#搜索引擎模式: 我/ 去过/ 清华/ 华大/ 大学/ 清华大学/ 和/ 北京/ 大学/ 北京大学/。

(3)方法实现

问题一中采用 jieba 分词先将训练集和测试集的文本分词，然后将分词后的获得的新文本存入一个新的文本文档，并建立文件夹(图 3-4)，方便后续函数调用。分词模式采用的是默认的精准分词模式.

▸cutResult = jieba.cut(result)　# 默认方式分词，分词结果用空格隔开(图 3-5)

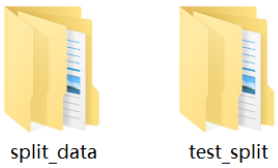


图 3-4 新的文件夹

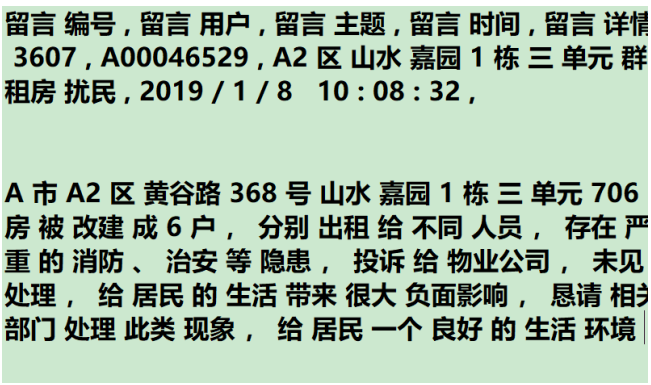


图 3-5 分词结果

3.2.3 TF-IDF 逆文本频率指数

TF-IDF(term frequency–inverse document frequency)[6]是一种用于信息检索与数据挖掘的常用加权技术。TF 意思是词频(Term Frequency)，IDF 意思是逆文本频率指数(Inverse Document Frequency)。主要用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。

TF-IDF 的主要思想是：如果某个词或短语在一篇文章中出现的频率 TF 高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。TFIDF 实际上是：TF * IDF，TF 词频(Term Frequency)，IDF 逆向文件频率(Inverse Document Frequency)。TF 表示词条在文档 d 中出现的频率。IDF 的主要思想是：如果包含词条 t 的文档越少，也就是 n 越小，IDF 越大，则说明词条 t 具有很好的类别区分能力。然而如果其他文档也出现比较多，说明该词条 t 区分性不大，就用 IDF 来降低该词条的权重。

词频(TF)的计算公式是：对于在某一特定文件里的词语t_i来说，它的重要性可表示为:

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

以上式子中$n_{i,j}$是该词在文件d_j中的出现次数，而分母则是在文件d_j中所有字词的出现次数之和，即

$$\mathrm{TF}_{\omega}=\frac{\text{在某一类中的词条 }\omega\text{ 出现的次数}}{\text{该类中所有的词条数目}}$$

逆向文件频率(inverse document frequency，IDF)是一个词语普遍重要性的度量。某一特定词语的 IDF，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取以 10 为底的对数得到:

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

即$\mathrm{IDF}=\log\left(\frac{\text{语料库的文档总数}}{\text{包括词条 }\omega\text{ 的文档数}+1}\right)$，分母之所以加 1，是为了避免分母为 0。

其中，$|D|$：语料库中的文件总数，$\left|\{j:t_i \in d_j\}\right|$：包含词语的文件数目(即文件数目)如果该词语不在语料库中，就会导致分母为零，因此一般情况下使用 $1+|\{d \in D:t \in d\}|$.

然后再计算 TF 与 IDF 的乘积

$$tfidf_{i,j} = tf_{i,j} \times idf_i$$

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。运用到问题一中具体步骤为首先调用已分词的文档，构建词频矩阵，滤过停用词，将计算好的 TF-IDF 向量提取出来(图 3-6).

```
(0, 13673)	0.1293050113024224
(0, 8661)	0.227009018358538
(0, 11013)	0.21287756167768196
(0, 8950)	0.13987186856837294
(0, 14229)	0.19507403178234348
(0, 4161)	0.20285113359937065
(0, 8967)	0.16759435274244577
(0, 628)	0.11482327568781425
(0, 16)	0.07700175953425002
(0, 207)	0.09208813788184525
(0, 416)	0.10163383474481805
(0, 6763)	0.4467474669184869
(0, 3510)	0.302553970458887
(0, 14539)	0.3434568249271633
```

图 3-6 TF-IDF 向量

3.2.4 朴素贝叶斯算法预测种类

(1)朴素贝叶斯算法建立的推导

我们通过学习与研究文献[7]知道，朴素贝叶斯分类模型是建立在贝叶斯定理上之的，所以我们需要从贝叶斯定理出发。

贝叶斯定理的简洁版：

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

其中，为了方便理解，当 B 出现作为 A 的条件出现时，我们假设它只有一个特征。但在实际应用中，很少有一件事只受一个特征影响的情况，那么假设影响 B 的因素有 n 个，分别是$b_1, b_2, \cdots, b_n$。

则上式$P(A|B)$可以写成：

$$P(A|b_1, b_2, \cdots, b_n) = \frac{P(A)P(b_1, b_2, \cdots, b_n|A)}{P(b_1, b_2, \cdots, b_n)}$$

其中，A 的先验概率 $P(A)$和多个因素的联合概率 $P(b_1, b_2, \cdots, b_n)$是可以单独计算的，所以此两项都可以被看作常数。关于求解$P(A|b_1, b_2, \cdots, b_n)$，关键在于$P(b_1, b_2, \cdots, b_n|A)$。由链式法则可得：

$$P(b_1, b_2, \cdots, b_n|A) = P(b_1|A)P(b_2|A, b_1)P(b_3|A, b_1, b_2)\cdots P(b_n|A, b_1, b_2, \cdots, b_{n-1})$$

假如，b_1 到 b_n 的这些特征在概率分布上是条件独立的，当$i \neq j$时，有$P(b_i|A, b_j) = P(b_i|A)$。因此，当$b_1, b_2, \cdots, b_n$中每个特征与其他$n-1$个特征都不相关时，有：

$$P(A|b_1, b_2, \cdots, b_n) = \frac{1}{Z}P(A)\prod_{i=1}^{n} P(bi|A)$$

其中，Z代表$P(b_1, b_2, \cdots, b_n)$。

(2)模型的建立

由上面的推导式可知，b_1到 b_n是代表特征(Feature)，A代表最终类别(Class)，据此我们可以得出朴素贝叶斯分类的模型函数：

$$P(C|F1, F2, \cdots, Fn) = \frac{1}{Z}P(A)\prod_{i=1}^{n} P(Fi|C)$$

(3)结果显示

```
./split/test_split/交通运输/113365901.txt :实际类别:  交通运输 -->预测类别:  城乡建设
./split/test_split/交通运输/1854891.txt :实际类别:  交通运输 -->预测类别:  劳动和社会保障
./split/test_split/交通运输/1856101.txt :实际类别:  交通运输 -->预测类别:  劳动和社会保障
./split/test_split/交通运输/1856481.txt :实际类别:  交通运输 -->预测类别:  城乡建设
./split/test_split/交通运输/1857161.txt :实际类别:  交通运输 -->预测类别:  商贸旅游
./split/test_split/交通运输/1858341.txt :实际类别:  交通运输 -->预测类别:  城乡建设
./split/test_split/交通运输/1859791.txt :实际类别:  交通运输 -->预测类别:  城乡建设
./split/test_split/卫生计生/3414701.txt :实际类别:  卫生计生 -->预测类别:  劳动和社会保障
./split/test_split/卫生计生/3429391.txt :实际类别:  卫生计生 -->预测类别:  城乡建设
./split/test_split/卫生计生/3432001.txt :实际类别:  卫生计生 -->预测类别:  劳动和社会保障
./split/test_split/卫生计生/3475731.txt :实际类别:  卫生计生 -->预测类别:  教育文体
./split/test_split/卫生计生/3499251.txt :实际类别:  卫生计生 -->预测类别:  劳动和社会保障
./split/test_split/城乡建设/1762851.txt :实际类别:  城乡建设 -->预测类别:  环境保护
./split/test_split/城乡建设/1776761.txt :实际类别:  城乡建设 -->预测类别:  劳动和社会保障
./split/test_split/城乡建设/1778801.txt :实际类别:  城乡建设 -->预测类别:  环境保护
./split/test_split/教育文体/3496421.txt :实际类别:  教育文体 -->预测类别:  劳动和社会保障
./split/test_split/教育文体/3500121.txt :实际类别:  教育文体 -->预测类别:  商贸旅游
./split/test_split/环境保护/1617391.txt :实际类别:  环境保护 -->预测类别:  教育文体
erroe rate: 8.61244019138756 %
```

图 3-7 一级模型分类结果显示错误率为 8.6124%

3.3 使用 F-Score 对模型进行评价

我们把数据分成七类(依次是城乡建设、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)，里面的数据是没分类的，然后用我们建立好的问题一分类模型做预测。所有拿出预测的数据中有 b 个第 i 类(在这里我们做测试的数据每类有 20 个)，m 表示预测出是第 i 类的个数(其中用 a 表示预测正确的个数)，P_i 表示第 i 类的查准率，R_i表示第 i 类的查全率，得到表 3-1。

表 3-1 各个指标值及 F 得分

i	b	m	a	P_i	R_i	F
1	20	32	20	20/32	20/20	0.7692
2	20	22	20	20/22	20/20	0.9524
3	20	33	14	14/33	14/20	0.6096
4	20	20	20	20/20	20/20	1
5	20	20	20	20/20	20/20	1
6	20	22	20	20/22	20/20	0.9524
7	20	20	20	20/20	20/20	1

最后我们将上表数据代入分类模型的评价指标 F-Score

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

结果算出

$$F_1 = \frac{1}{7}\sum_{i=1}^{7}\frac{2P_iR_i}{P_i + R_i} = 0.8977$$

从用 F-Score 对模型评价的结果可以看出:我们的模型对交通运输类的预测准确率较低，对其他的类型的准确率很高，整体来看我们的分类预测属于良好。说明基于朴素贝叶斯算法的文本分类模型对群众留言文本分类总体有有较高的准确性。

四、问题二的模型建立与求解

4.1 信息抽取技术

信息抽取技术[10]是计算机识别领域主题的基础性课题。目前主题抽取方法归纳起来可分为三类：统计学方法、语言学方法以及统计学和语言学相融合的方法，三种方法都有优缺点。本题可以采用 TF-IDF，即公式(1)的算法结合主题词表的方法进行信息抽取。该方法由 TF-IDF 算法结果值的基础词表和领域词表组成，通过固定领域词表中主题 TF-IDF 权值，增加了领域主题的独占性的计算，筛选出领域主题词。在抽取出的主题词队列中，对主题词权重与主题词长度进行乘法计算，人工评价结果明显提高。

$$W(t,d)=\frac{tf(t,d)\times\log\left(\frac{N}{n_i}+0.01\right)}{\sqrt{\sum_{t\in d}\{tf(t,d)\times\log\left(\frac{N}{n_i}+0.01\right)\}^2}} \qquad \text{公式(1)}$$

4.2 语义相似度统计

Word2vec(图 4-1)是构建多数层的神经网络模型，word2vec 是构建多层的神经网络模型，然后给定输入和输出求出相应的相似度。word2vec 基础算法是 N-gram，所以 n 元模型中如果在不改变词语在上下文中的顺序前提下，距离相近的词语关系越近。距离较远的关联度越远，当距离足够远时，词语之间则没有关联度。常用的 NLP 的 python 模块有 gensim，NLTK 等。本文使用 gensim 模块。

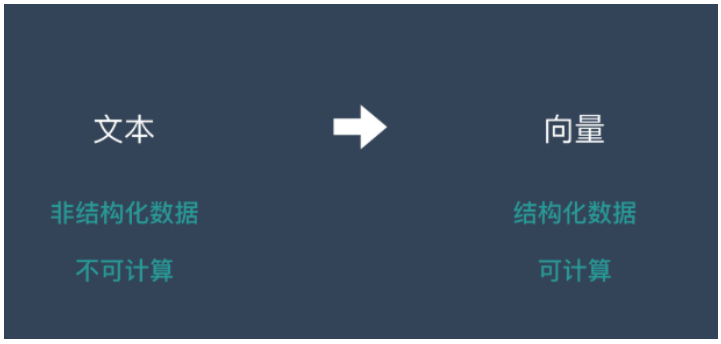


图 4-1 word2vec 算法

4.2.1 训练前语料处理

第一步、分词：如问题一知进行分词，分词工具使用的是 jieba，我们分别使用 jieba 直接进行分词和使用自定义词典作为词库来分词，来保证分词的准确性。停用词的概念：这些功能词的两个特征促使在搜索引擎的文本处理过程中对其特殊对待。第一，这些功能词极其普遍。记录这些词在每一个文档中的数量需要很大的磁盘空间。第二，由于它们的普遍性和功能，这些词很少单独表达文档相关程度的信息。如果在检索过程中考虑每一个词而不是短语，这些功能词基本没有什么帮助。在信息检索中，这些功能词的另一个名称是:停用词(stopword)。称它

们为停用词是因为在文本处理过程中如果遇到它们，则立即停止处理，将其扔掉。将这些词扔掉减少了索引量，增加了检索效率，并且通常都会提高检索的效果。停用词主要包括英文字符、数字、数学字符、标点符号及使用频率特高的单汉字等。python 中有 stopword.txt(与问题一同)。

分词结果如图显示：

['留言', '主题', '咨询', '区', '道路', '命名', '规划', '初步', '成果', '公示',

第二步、模型训练：

使用 word2vec 每次的结果都不一样，原因是模型不用重新训练。第一次实例没有使用自定义的词典。

训练 Word2Vec 的思想，是利用一个词和它在文本中的上下文的词，这样就省去了人工去标注。

CBOW 用环境词预测中心词(图 4-2)，得到逻辑回归网络可以用来预测类似“一句话中缺少了一个单词，这个单词最可能是什么”这样的问题[11]。

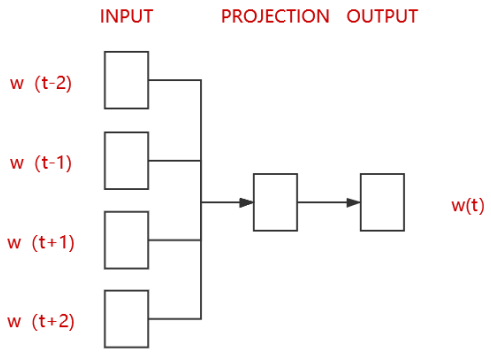


图 4-2 环境词预测中心词

4.2.2 LDA 主题模型的构建实例

在文本挖掘领域，大量的数据都是非结构化的，很难从信息中直接获取相关和期望的信息，一种文本挖掘的方法：主题模型(Topic Model)能够识别在文档里的主题，并且挖掘语料里隐藏信息，并且在主题聚合、从非结构化文本中提取信息、特征选择等场景有广泛的用途。

主题可以被定义为“语料库中具有相同词境的词的集合模式”，比如说，主题模型可以

将“学习”，“学校”，“书本”，“宿舍” 集合成 “学校生活” 主题

将 “口红”，“眉笔”，“腮红” 集合成 “化妆品”主题

LDA(Latent Dirichlet Allocation)是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。所谓生成模型，就是说，我们认为一篇文章的每个词都是通过“以一定概率选择了某个主题，并从这个主题中

以一定概率选择某个词语”这样一个过程得到。文档到主题服从多项式分布，主题到词服从多项式分布。

LDA 是一种非监督机器学习技术，可以用来识别大规模文档集(document collection)或语料库(corpus)中潜藏的主题信息。它采用了词袋(bag of words)的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

[dic.doc2bow(text) for text in cut_file]就是要生成相应的字典，抽取文档中的单词组成一个字典。

gensim.models.ldamodel.LdaModel(corpus =corpus_tfidf, id2word = dictionary, num_topics = 50, update_every=1, chunksize=100,passes=1)

对于 model 的那个构造函数的解释：alpha 是一个重要的参数，较大的 alpha 值会导致每个文档中包含更多的主题，alpha 必须是正数，通常很小，一般小于 1,默认是 1.0/len(corpus)。这时 topics 就是目前分出来的主题，主题模型是一个稀疏的模型，即便每个文档中有很多潜在主题，也只有一小部分会被用到。

4.3 热点评价指数

由于本文需要通过关键词的热度来进一步确定当前的热点话题，故结合主题的对于关键词的提及数和主题的点赞数对关键词权重作进一步改进。具体公式如下：

$$p = \frac{s_i}{\sqrt{\sum {s_i}^2}} + \frac{z_i}{\sqrt{\sum {z_i}^2}}$$

其中，$z =$ 点赞数，$s =$ 提及数，热度 $=$ score $*$ p $*$ 1000，score $=$ 关键词权重。

4.4 结果显示

最后表格部分展示如图 4-3 和图 4-4 所示：

A	B	C	D	E	F
热度排名	问题ID	热度指数	时间范围	地点/人群	问题描述
1	1	0.154	2019/01/01至2020/01/07	A市A7县居民	A7县镇上居民反映交通和施工扰民等问题
2	2	0.102	2019/01/07至2020/01/07	A市A3区居住的人们	A3区举报扰民和安全隐患等问题
3	3	0.096	2019/02/11至2020/01/07	A市A2区小区居住者	A市A2区居民投诉附近有噪音等扰民问题

图 4-3 热点问题表

A	B	C	D	E	F	G	H
4	199446	A00027173	语实验学校急需要解决公	2019/4/16 15:08:09	，却需要坐两趟公交车	0	2
4	244099	A00073143	园后面建学校堵塞了小区	2019/4/14 10:20:03	无就业安置，山田水土	0	0
4	270174	A00080995	决A4区青竹湖片区的学校	2019/3/3 19:36:39	语学校和楚一立信实验	0	0
4	188467	A00050188	市温斯顿英语培训学校拖延	2019/3/28 19:57:19	校都是推辞的态度！去	0	1
4	255744	A00058664	院14栋社区艾瑞弗艺术培	2019/3/21 21:21:18	的群众哭哭啼啼的求告	0	1
4	252212	A00078227	市直学校教师招聘考试不	2019/3/21 11:33:48	；30到达学校，迟到了	0	1
4	259696	A00013333	北大培文学校后期如何规	2019/3/2 14:35:27	民办学校，旨在将学校	0	0
4	203208	A00043904	A5区新华都学校西南角红	2019/2/25 23:00:53	走，直行灯变绿时，左	0	0
4	224239	A00029915	市外国语学校周日收费补	2019/2/24 19:35:26	父家长群接龙报名、收费	0	0
4	237413	A00010614	临时老师的发放金额为何	2019/2/21 19:50:03	放的金额，是区教育政府	0	0
4	264240	A00039671	长郡月亮岛学校附近噪音很	2019/2/21 17:00:02	渣土车，就是那种带	0	0
4	200013	A00031618	中昊城实验学校周边交通	2019/2/2 18:01:47	事故频发，学生是国家	0	1

图 4-4 热点问题留言明细表

五、问题三的模型建立

问题三的评价方案主要需要建立一个文本相关度计算的模型，这里我们参照基于在线百科知识库的文本语义相关度计算[12]，建立模型。

5.1 模型介绍

模型的观点即以维基百科中的一个主题为一个语义概念，该概念同时携带有一篇描述该概念的大段文本。本文假定从中文维基百科中抽取出 n 个语义概念，由这 n 个概念可以构成一个 1"1 维的语义概念空间。而任意一段待处理文本都可将其表示为该空间中的一点，即本文的文本语义表示模型。两个文本的语义相关度计算则转换为空间中两问的距离。

5.2 基于维基百科的文本语义表示模型

文本语义相关度计算的核心问题在于文本语义如何表示，再对二者的文本语义内涵进行相关性度量。接下来，首先讲述本文的文本语义表示模型。

本问模型从概念的相关性角度来表示文本的语义。假定有一个基本概念组成的向量$(c_1,\cdots,c_i,\cdots,c_n)$，则任意一个文本片段 t 的语义可由一个权值向量$(w_1,\cdots,w_i,\cdots,w_n)$来表示。其中，$W_j$是 t 与C_i之间的语义相关性强度。即，将基本概念集视为一个 n 维语义空间，而每个文本 t 的语义就对应于 n 维空间中的一点，t 的语义就是 t 与每一维基本概念的相关性强度组成的向量$(w_1,\cdots,w_i,\cdots,w_n)$。

将文本转化为 n 维空间中的一点后，两个文本之间的语义相关性计算就等价于空间中两点距离的计算，而两点间距离的计算可以采用多种成熟的计算方法。

下面，讲述该文本语义表示模型中的两个核心问题：基本概念集的构建和文本语义表示算法。

5.2.1 基本概念集的构建

为了使该文本语义表示模型能够应用于 NLP 领域的多个任务中，并且避免人工构建词汇知识库的缺点，该模型的核心问题——基本概念集合应该至少满足以下几个条件：

①尽可能多地覆盖到目前人类认知的各种主题知识；

②容易维护，新概念的加入要方便；

③这些概念要容易被人类所理解和使用，同时，由其表示出的文本语义也是人类通俗易懂的；

④每个概念C_i应该伴随有对其作出解释的文本内容d_i，通过d_i可以计算出每个待处理文本 t 与C_i的语义相关性。

5.2.2 文本语义表示算法

假定有一个概念集合$(c_1,\cdots,c_i,\cdots,c_n)$，和一个与该概念集合相关联的文档集$(d_1,\cdots,d_i,\cdots,d_n)$，我们来构建一个矩阵 T，T 中包含 n 列，每列对应一个概念，每行对应一个词语，该词语是来自于文档集中所有不重复的词语$\cup_{i=1\cdots n}\ d_i$。$T[i,j]$表示词语在文档中的 TFIDF 值，如以下公式：

$$T[i,j]=tf(t_i,d_i)\times log\frac{n}{df_i}$$

其中,当$count(t_i,d_j)>0$时，$tf(t_i,d_j)=1+\log count\ (t_i,d_j)$

否则，$tf(t_i,d_j)=0$，而$df_i=|\{d_k:t_i\in d_k\}|$，即文档集中包含词语$t_i$的文档个数(文档频率)。

最后，对 TFIDF 值进行归一化处理。

$T(i,j)\leftarrow\frac{T[i,j]}{\sqrt{\Sigma_{i=1}^{r}T[i,j]^2}}$ 其中，r是词语的总个数.

这样，词语t的语义即为T中第i行组成的向量，即$T[i,j]$为t_i与每个基本概念的语义相关度。T实为加权的倒排索引矩阵。

5.3 文本语义相关度计算

本文的方法可以将文本表示为高维空间中的语义向量，而计算文本相关度就等价于比较向量。而向量的比较可以用很多种成熟的度量方法，本文采用余弦距离。图 5-1 给出本文的系统实现流程。

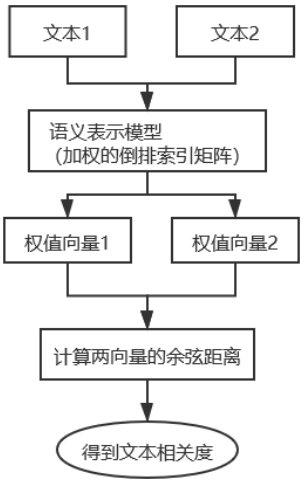


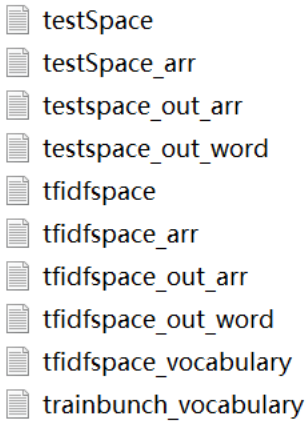
图 5-1 系统实现流程

六、模型评价

6.1 问题一模型评价

6.1.1 模型优点

朴素贝叶斯分类模型是目前公认的一种简单有效的概率分类方法，在文本分类方面表现出很好的性能，它对缺失数据不是特别敏感，算法简单易操作，而且比其他分类方法的准确率更高。此外，在朴素贝叶斯分类的方法中，有一个“独立性假设”即留言文本之间的每个特征相互独立，这使得朴素贝叶斯法特别适合处理特征有很多的分类任务。恰好这次我们需要的是将网络问政平台上的群众留言进行分类，而留言文本是属于多特征的分类任务。



以上 10 个 txt 文件，由上到下分别是：

testSpace.txt #测试集分词信息
testSpace_arr.txt #测试集词频矩阵信息
testspace_out_arr.txt #测试集输出矩阵信息
testspace_out_word.txt #测试界单词信息
tfidfspace.txt #将 TF-IDF 词向量保存为 txt，方便查看

tfidfspace_arr.txt #将 TF-IDF 词频矩阵保存为 txt，方便查看
tfidfspace_out_arr.txt #tfidf 输出矩阵信息
tfidfspace_out_word.txt #单词形式的 txt
tfidfspace_vocabulary.txt #将分词的词汇统计信息保存为 txt，方便查看
trainbunch_vocabulary.txt #所有分词词频信息
是 NLP(自然语言处理)非常珍贵的资源

6.1.2 模型缺点

对于一些复杂语法结构的留言文本进行分类还是不太准确。朴素贝叶斯分类的前提较理性化，现实生活中语言文本的特征之间相互独立的假设不一定成立，如果文本的特征之间的相关较大，该模型分类的准确率将会下降。

6.2 问题二模型评价

6.1.1 模型的优点

TF-IDF 算法实现简单快速，算法简单。对于初学者等使用和操作方便易懂。

Word2vec 测试相似度效果比较准确，而且易操作，而且由于 word2vec 会考虑上下文，跟 Embedding 方法相比，效果要更好。比之前的的 embedding 方法维数更少，所以速度更快。通用性很强，可以用在各种 NLP 任务中。

6.1.2 模型的不足

TF-IDF 采用文本逆频率 IDF 对 TF 值加权取权值大的作为关键词，但 IDF 的简单结构并不能有效地反映单词的重要程度和特征词的分布情况，使其无法很好地完成对权值调整的功能，所以 TF-IDF 算法的精度并不是很高，尤其是当文本集已经分类的情况下。

Word2vec 由于词向量是一对一的关系，所以多义词的问题无法解决。

6.3 问题三模型评价

问题三运用的是基于在线百科知识库的文本语义相关与计算的模型。该模型在中文维基百科知识库的基础上，对文本语义相关度计算进行了研究。实验选取了 2014 年 12 月 15 日在中文维基百科网站下载的主题文章，进行处理后作为语义概念知识库。在 words 一 240 测试集上的实验结果表明，该方法比基于 WordNet 的 LsA 算法的效果要好。

该方法的优点主要有两个：一者，该知识库包含的知识要比人工构建的词汇知识库大很多，并且是以人类的自然语言进行描述；二者，该方法可以用于 NLP 领域的很多任务，如文本分类、聚类、相关度、相似度计算等。

参考文献

[1]张涛，邓兆安，中国式网络问政:”胶东在线‘的标本意义，南方日报出版社，2010.

[2]倪晓峰.全省最大的网络问政平台——问政江西 就在您身边 中国江西网 2019-01-15.

[3]周健民．土壤学大辞典：科学出版社，2013.10.

[4]付泉．管理信息系统：华中科技大学出版社，2013.

[5]石洪波, 王志海, 黄厚宽. 贝叶斯文本分类方法研究[J]. 高等财经教育研究, 2002, 000(0S1):P.87-88.

[6] CSDN 博主「海涛 anywn」 TF-IDF 介绍及应用.

https://blog.csdn.net/lihaitao000/article/details/51307365.

[7]王艺颖. 朴素贝叶斯方法在中文文本分类中的应用[J]. 中国高新科技, 2019, 43(07):59-62.

[8]袁 方, 苑俊英. 基于类别核心词的朴素贝叶斯中文文本分类[J]. 山东大学学报 (理学版), 2006, 41(3):111-114.

[9]Mccallum A, Nigam K. A comparison of event models for naive bayes text classification[C]. national conference on artificial intelligence, 1998: 41-48.

[10] 施韶亭，曹方.文本挖掘技术在科技管理领域热点主题抽取方向的应用研究.

[11] CSDN 博主[To_be_brave1] word2vec and glove 优缺点.

https://blog.csdn.net/u012879957/article/details/82735057.

[12]刘海静. 基于在线百科知识库的文本语义相关度计算[J]. 洛阳师范学院学报, 2015(05):86-89.

附录

附录 1

```
    def readFile(path):
    with open(path, 'r', errors='ignore') as file:
    # 文档中编码有些问题，所有出现问题的用 errors 过滤错误
        content = file.read()
        file.close()
        return content


def saveFile(path, result):
    with open(path, 'w', errors='ignore') as file:
        file.write(result)
        file.close()


def segText(inputPath, resultPath):
    fatherLists = os.listdir(inputPath)  # 这个为主目录
    for eachDir in fatherLists:  # 遍历主目录中各个文件夹
        eachPath = inputPath + eachDir + "/"  # 保存主目录中每个文件夹的目录，方便遍历二级文件
        each_resultPath = resultPath + eachDir + "/"  # 这个是分词结果文件存入的目录
        if not os.path.exists(each_resultPath):
            os.makedirs(each_resultPath)
        childLists = os.listdir(eachPath)  # 获取每个文件夹中的各个文件
        for eachFile in childLists:  # 遍历每个文件夹中的子文件
            eachPathFile = eachPath + eachFile  # 获得每个文件路径
            #  print(eachFile)
            content = readFile(eachPathFile)  # 调用上面函数读取文件里的内容
            # content = str(content)
            result = (str(content)).replace("\r\n", "").strip()  # 删除多余空行与空格
            # result = content.replace("\r\n","").strip()

            cutResult = jieba.cut(result)  # 默认方式的、词，分词结果用空格隔开
            saveFile(each_resultPath + eachFile, " ".join(cutResult))  # 调用上面函数保存文件


def bunchSave(inputFile, outputFile):
    catelist = os.listdir(inputFile)
    bunch = Bunch(target_name=[], label=[], filenames=[], contents=[])
```

```
    bunch.target_name.extend(catelist)  # 将类别保存到 Bunch 对象中
    for eachDir in catelist:
        eachPath = inputFile + eachDir + "/"
        fileList = os.listdir(eachPath)
        for eachFile in fileList:  # 二级目录中的每个子文件
            fullName = eachPath + eachFile  # 二级目录子文件全路径
            bunch.label.append(eachDir)  # 当前分类标签
            bunch.filenames.append(fullName)  # 保存当前文件的路径
            bunch.contents.append(readFile(fullName).strip())  # 保存文件词向量
    with open(outputFile, 'wb') as file_obj:  # 持久化必须用二进制访问模式打开
        pickle.dump(bunch, file_obj)
        # pickle.dump(obj, file, [,protocol])函数的功能：将 obj 对象序列化存入已经打开的
file 中。
        # obj：想要序列化的 obj 对象。
        # file:文件名称。
        # protocol：序列化使用的协议。如果该项省略，则默认为 0。如果为负值或
HIGHEST_PROTOCOL，则使用最高的协议版本


def readBunch(path):
    with open(path, 'rb') as file:
        bunch = pickle.load(file)
        # pickle.load(file)
        # 函数的功能：将 file 中的对象序列化读出。
    return bunch


def writeBunch(path, bunchFile):
    with open(path, 'wb') as file:
        pickle.dump(bunchFile, file)


def getStopWord(inputFile):
    stopWordList = readFile(inputFile).splitlines()
    return stopWordList


def getTFIDFMat(inputPath, stopWordList, outputPath,
                    tftfidfspace_path,tfidfspace_arr_path,tfidfspace_vocabulary_path):  # 求得
TF-IDF 向量
    bunch = readBunch(inputPath)
    tfidfspace = Bunch(target_name=bunch.target_name, label=bunch.label,
filenames=bunch.filenames, tdm=[],
                       vocabulary={})
```

```
    '''读取 tfidfspace'''
    tfidfspace_out = str(tfidfspace)
    saveFile(tftfidfspace_path, tfidfspace_out)
    # 初始化向量空间
    vectorizer = TfidfVectorizer(stop_words=stopWordList, sublinear_tf=True, max_df=0.5)
    transformer = TfidfTransformer()  # 该类会统计每个词语的 TF-IDF 权值
    # 文本转化为词频矩阵，单独保存字典文件
    tfidfspace.tdm = vectorizer.fit_transform(bunch.contents)
    tfidfspace_arr = str(vectorizer.fit_transform(bunch.contents))
    saveFile(tfidfspace_arr_path, tfidfspace_arr)
    tfidfspace.vocabulary = vectorizer.vocabulary_  # 获取词汇
    tfidfspace_vocabulary = str(vectorizer.vocabulary_)
    saveFile(tfidfspace_vocabulary_path, tfidfspace_vocabulary)
    '''over'''
    writeBunch(outputPath, tfidfspace)


def getTestSpace(testSetPath, trainSpacePath, stopWordList, testSpacePath,
                 testSpace_path,testSpace_arr_path,trainbunch_vocabulary_path):
    bunch = readBunch(testSetPath)
    # 构建测试集 TF-IDF 向量空间
    testSpace = Bunch(target_name=bunch.target_name, label=bunch.label,
filenames=bunch.filenames, tdm=[],
                      vocabulary={})
    '''
        读取 testSpace
        '''
    testSpace_out = str(testSpace)
    saveFile(testSpace_path, testSpace_out)
    # 导入训练集的词袋
    trainbunch = readBunch(trainSpacePath)
    # 使用 TfidfVectorizer 初始化向量空间模型  使用训练集词袋向量
    vectorizer = TfidfVectorizer(stop_words=stopWordList, sublinear_tf=True, max_df=0.5,
                                 vocabulary=trainbunch.vocabulary)
    transformer = TfidfTransformer()
    testSpace.tdm = vectorizer.fit_transform(bunch.contents)
    testSpace.vocabulary = trainbunch.vocabulary
    testSpace_arr = str(testSpace.tdm)
    trainbunch_vocabulary = str(trainbunch.vocabulary)
    saveFile(testSpace_arr_path, testSpace_arr)
    saveFile(trainbunch_vocabulary_path, trainbunch_vocabulary)
    # 持久化
    writeBunch(testSpacePath, testSpace)
```

```
def bayesAlgorithm(trainPath, testPath,tfidfspace_out_arr_path,
                           tfidfspace_out_word_path,testspace_out_arr_path,
                           testspace_out_word_apth):
    trainSet = readBunch(trainPath)
    testSet = readBunch(testPath)
    clf = MultinomialNB(alpha=0.001).fit(trainSet.tdm, trainSet.label)
    # alpha:0.001 alpha  越小，迭代次数越多，精度越高
    # print(shape(trainSet.tdm))  #输出单词矩阵的类型
    # print(shape(testSet.tdm))
    '''处理 bat 文件'''
    tfidfspace_out_arr = str(trainSet.tdm)   #  处理
    tfidfspace_out_word = str(trainSet)
    saveFile(tfidfspace_out_arr_path, tfidfspace_out_arr)   #  矩阵形式的 train_set.txt
    saveFile(tfidfspace_out_word_path, tfidfspace_out_word)   #  文本形式的 train_set.txt

    testspace_out_arr = str(testSet)
    testspace_out_word = str(testSet.label)
    saveFile(testspace_out_arr_path, testspace_out_arr)
    saveFile(testspace_out_word_apth, testspace_out_word)

    '''处理结束'''
    predicted = clf.predict(testSet.tdm)
    total = len(predicted)
    rate = 0
    for flabel, fileName, expct_cate in zip(testSet.label, testSet.filenames, predicted):
        if flabel != expct_cate:
            rate += 1
            print(fileName, ":实际类别： ", flabel, "-->预测类别： ", expct_cate)
    print("erroe rate:", float(rate) * 100 / float(total), "%")



# 分词，第一个是分词输入，第二个参数是结果保存的路径

#
if __name__ == '__main__':
    datapath = "./data/"   #原始数据路径
    stopWord_path = "./stop/stopword.txt"#停用词路径
    test_path = "./test/"                  #测试集路径
    '''
    以上三个文件路径是已存在的文件路径，下面的文件是运行代码之后生成的文件路径
    dat 文件是为了读取方便做的，txt 文件是为了给大家展示做的，所以想查看分词，词频
矩阵
```

词向量的详细信息请查看 txt 文件，dat 文件是通过正常方式打不开的

'''

test_split_dat_path = "./test_set.dat" #测试集分词 bat 文件路径

testspace_dat_path ="./testspace.dat" #测试集输出空间矩阵 dat 文件

train_dat_path = "./train_set.dat" # 读取分词数据之后的词向量并保存为二进制文件

tfidfspace_dat_path = "./tfidfspace.dat" #tf-idf 词频空间向量的 dat 文件

'''

以上四个为 dat 文件路径，是为了存储信息做的，不要打开

'''

test_split_path = './split/test_split/' #测试集分词路径

split_datapath = "./split/split_data/" # 对原始数据分词之后的数据路径

'''

以上两个路径是分词之后的文件路径，大家可以生成之后自行打开查阅学习

'''

tfidfspace_path = "./tfidfspace.txt" # 将 TF-IDF 词向量保存为 txt，方便查看

tfidfspace_arr_path = "./tfidfspace_arr.txt" # 将 TF-IDF 词频矩阵保存为 txt，方便查看

tfidfspace_vocabulary_path = "./tfidfspace_vocabulary.txt" # 将分词的词汇统计信息保存为 txt，方便查看

testSpace_path = "./testSpace.txt" #测试集分词信息

testSpace_arr_path = "./testSpace_arr.txt" #测试集词频矩阵信息

trainbunch_vocabulary_path = "./trainbunch_vocabulary.txt" #所有分词词频信息

tfidfspace_out_arr_path = "./tfidfspace_out_arr.txt" #tfidf 输出矩阵信息

tfidfspace_out_word_path = "./tfidfspace_out_word.txt" #单词形式的 txt

testspace_out_arr_path = "./testspace_out_arr.txt" #测试集输出矩阵信息

testspace_out_word_apth ="./testspace_out_word.txt" #测试界单词信息

'''

以上 10 个文件是 dat 文件转化为 txt 文件，大家可以查询信息，这是 NLP(自然语言处理)非常珍贵的资源

'''

#输入训练集

segText(datapath,#读入数据

split_datapath)#输出分词结果

bunchSave(split_datapath,#读入分词结果

train_dat_path) # 输出分词向量

stopWordList = getStopWord(stopWord_path) # 获取停用词表

getTFIDFMat(train_dat_path, #读入分词的词向量

stopWordList, #获取停用词表

tfidfspace_dat_path, #tf-idf 词频空间向量的 dat 文件

tfidfspace_path, #输出词频信息 txt 文件

tfidfspace_arr_path,#输出词频矩阵 txt 文件

tfidfspace_vocabulary_path) #输出单词 txt 文件

'''

测试集的每个函数的参数信息请对照上面的各个信息，是基本相同的

```
    '''
    #输入测试集
    segText(test_path,
            test_split_path)   # 对测试集读入文件，输出分词结果
    bunchSave(test_split_path,
              test_split_dat_path)   #
    getTestSpace(test_split_dat_path,
                 tfidfspace_dat_path,
                 stopWordList,
                 testspace_dat_path,
                 testSpace_path,
                 testSpace_arr_path,
                 trainbunch_vocabulary_path)# 输入分词文件，停用词，词向量，输出特征空间(txt,dat 文件都有)
    bayesAlgorithm(tfidfspace_dat_path,
                   testspace_dat_path,
                   tfidfspace_out_arr_path,
                   tfidfspace_out_word_path,
                   testspace_out_arr_path,
                   testspace_out_word_apth)
```

附录 2

```
    # -*- coding: utf-8 -*-
import jieba
import re
from collections import Counter
import codecs
stop_words = 'stopword.txt'
stopwords = codecs.open(stop_words, 'r', encoding='utf8').readlines()
stopwords = [w.strip() for w in stopwords]
stop_flag = ['x', 'c', 'u', 'd', 'p', 't', 'uj', 'm', 'f', 'r']

stop_words=""
for line in open('./1720.txt',encoding='utf-8'):
    line.strip('\n')
    line = re.sub("[A-Za-z0-9\：\·\—\，\。\“ \”]", "", line)
    seg_list=jieba.cut(line,cut_all=False)
```

```
        stop_words+=(" ".join(seg_list))
all_words=stop_words.split()
print(all_words)
c=Counter()
for x in all_words:
        if len(x)>1 and x != '\r\n':
                c[x] += 1

print('\n 词频统计结果：')
```

```
        # -*- coding: utf-8 -*-
import jieba.posseg as pseg
import jieba.analyse
import xlwt
import openpyxl
from gensim import corpora, models, similarities
import re

#停词函数
def StopWordsList(filepath):
        wlst = [w.strip() for w in open(filepath, 'r', encoding='utf8').readlines()]
        return wlst

def str_to_hex(s):
        return ''.join([hex(ord(c)).replace('0x', '') for c in s])

# jieba 分词
def seg_sentence(sentence, stop_words):
        stop_flag = ['x', 'c', 'u', 'd', 'p', 't', 'uj', 'f', 'r']
        sentence_seged = pseg.cut(sentence)
        outstr = []
        for word, flag in sentence_seged:
                if word not in stop_words and flag not in stop_flag:
                        outstr.append(word)
        return outstr

if __name__ == '__main__':
        #1  这些是 jieba 分词的自定义词典
```

```
#2 停用词，。
spPath = 'stopword.txt'
stop_words = StopWordsList(spPath)

#3 excel 处理
wbk = xlwt.Workbook(encoding='ascii')
sheet = wbk.add_sheet("sheet1")  # sheet 名称
sheet.write(0, 0, '1')
sheet.write(0, 1, '2')
sheet.write(0, 2, 'sheet')
wb = openpyxl.load_workbook('f.xlsx')
ws = wb.active
col = ws['B']
# 4 相似性处理
rcount = 1
texts = []
orig_txt = []
key_list = []
name_list = []
sheet_list = []

for cell in col:
    if cell.value is None:
        continue
    if not isinstance(cell.value, str):
        continue
    item = cell.value.strip('\n\r').split('\t')  # 将制表格切分
    string = item[0]
    if string is None or len(string) == 0:
        continue
    else:
        textstr = seg_sentence(string, stop_words)
        texts.append(textstr)
        orig_txt.append(string)
dictionary = corpora.Dictionary(texts)
feature_cnt = len(dictionary.token2id.keys())
corpus = [dictionary.doc2bow(text) for text in texts]
tfidf = models.LsiModel(corpus)
index = similarities.SparseMatrixSimilarity(tfidf[corpus], num_features=feature_cnt)
result_lt = []
word_dict = {}
count =0
for keyword in orig_txt:
    count = count+1
```

```
            print('开始执行，第'+ str(count)+'行')
            if keyword in result_lt or keyword is None or len(keyword) == 0:
                continue
            kw_vector = dictionary.doc2bow(seg_sentence(keyword, stop_words))
            sim = index[tfidf[kw_vector]]
            result_list = []
            for i in range(len(sim)):
                if sim[i] > 0.5:
                    if orig_txt[i] in result_lt and orig_txt[i] not in result_list:
                        continue
                    result_list.append(orig_txt[i])
                    result_lt.append(orig_txt[i])
            if len(result_list) >0:
                word_dict[keyword] = len(result_list)
            if len(result_list) >= 1:
                sname = re.sub(u"([^\u4e00-\u9fa5\u0030-\u0039\u0041-\u005a\u0061-\u007a])",
"", keyword[0:10])+ '_'\
                        + str(len(result_list)+ len(str_to_hex(keyword))) + str_to_hex(keyword)[-5:]
                sheet_t = wbk.add_sheet(sname)    # Excel 单元格名字
                for i in range(len(result_list)):
                    sheet_t.write(i, 0, label=result_list[i])

        #5  按照热度排序
        with open("rjlw.txt", 'w', encoding='utf-8') as wf2:
            orderList = list(word_dict.values())
            orderList.sort(reverse=True)
            count = len(orderList)
            for i in range(count):
                for key in word_dict:
                    if word_dict[key] == orderList[i]:
                        key_list.append(key)
                        word_dict[key] = 0
            wf2.truncate()
        #6  写入目标 excel
        for i in range(len(key_list)):
            sheet.write(i+rcount, 0, label=key_list[i])
            sheet.write(i+rcount, 1, label=orderList[i])
            if orderList[i] >= 1:
                shname = re.sub(u"([^\u4e00-\u9fa5\u0030-\u0039\u0041-\u005a\u0061-\u007a])",
"", key_list[i][0:10]) \
                         + '_'+ str(orderList[i]+ len(str_to_hex(key_list[i])))+
str_to_hex(key_list[i])[-5:]
                link = 'HYPERLINK("#%s!A1";"%s")' % (shname, shname)
                sheet.write(i+rcount, 2, xlwt.Formula(link))
```

```
rcount = rcount + len(key_list)
key_list = []
orderList = []
texts = []
orig_txt = []
wbk.save('jieguo.xls')
```