“智慧政务”中的文本挖掘应用

摘要：

近年来，随着微信、微博、市长邮箱、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、云计算、人工智能等技术的发展，建立基于自然语言处理技术的智慧政务系统已经是社会治理创新发展的新趋势，对提升政府的管理水平和施政效率具有极大的推动作用。

在处理网络问政平台的群众留言时，工作人员首先按照一定的划分体系，如内容分类三级标签体系，对留言进行分类，以便后续将群众留言分派至相应的职能部门处理。目前，大部分电子政务系统还是依靠人工根据经验处理，存在工作量大、效率低，且差错率高等问题。

本文建立了对群众留言内容的一级标签分类模型，并将某一时段内反映特定地点或特定人群问题的留言进行了归类，定义出合理的热度评价指标，从而得到排名前五的热点问题及其对应的留言信息，最后针对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做出一套评价方案。

本文第一章简单介绍本题背景，并说明文章应解决的问题。第二章对所使用的算法与函数进行了简要介绍与分析。在第三章中详细地介绍了处理问题的思路与具体流程：对于群众留言的分类模型，首先使用了 jieba 分词法生成词云，再利用 TF-IDF 提取关键词，最后使用 LDA 建模及贝叶斯算法分类，建立一级标签分类模型；对于热点问题的挖掘，使用 TF-IDF 方法及 Gensim 技术对留言进行提取，定义了热度评价指标，从而得到热点问题；对于留言的答复意见，通过对留言答复意见的各个角度的需求，设计出合理的评价方案。

本文所涉及的模型是通过对数据的多次尝试后得到的，建立的分类模型可以较方便地对群众留言进行归类，并且能够根据热度评价指标提取出热点问题，同时对答复意见得出合理评价。

关键词：jieba 分词　TF-IDF 朴素贝叶斯 LSI（潜语义）分析模型

目录

1.挖掘目标

1.1 挖掘背景

目前，人类社会已经进入了“大数据时代”，在大数据时代，数据是无所不在、无所不能被感知、提取和记录、无所不能被传输、无所不能被挖掘分析利用、无所不能被追溯和在线的。大数据背后隐含着巨大的社会、经济与科研价值，并已经在教育、医疗、经济等领域得以应用。许多国家已将大数据的建设和发展提升为国家战略，积极推动大数据相关应用的进展，为了更好地满足人们在大数据时代下的需求以及更好地为人们提供服务，政府管理者需要借助大数据相关技术，发挥大数据的优势进行职能、管理方式及服务方式的转变。但由于政府工作响应的延迟性以及工作系统的复杂性，政府系统的运营与管理面临着极大的挑战与转型冲击。

近年来，随着微信、微博、市长邮箱、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、云计算、人工智能等技术的发展，建立基于自然语言处理技术的智慧政务系统已经是社会治理创新发展的新趋势，对提升政府的管理水平和施政效率具有极大的推动作用。

在处理网络问政平台的群众留言时，工作人员首先按照一定的划分体系，如内容分类三级标签体系，对留言进行分类，以便后续将群众留言分派至相应的职能部门处理。目前，大部分电子政务系统还是依靠人工根据经验处理，存在工作量大、效率低，且差错率高等问题。

1.2 挖掘意义

近年来，人们的政治活动逐渐转移到网络空间，社会公共事务交流与传播愈发网络化，网络成为了政治活动的重要场地和媒介。政府也日益借助于互联网加强与人民群众的互动，及时地了解人民群众的关注焦点与反馈建议等。网络问政反映的是公众根本利益诉求的表达，这种表达其实是一种信息传递过程，公众将信息传递给政府，并希望得到政府的关注和回复，网络问政的正常、高效运行对

于拓宽公众参政议政渠道、积极解决公众的各项诉求、建设更高效的服务型政府都具有非常重要的意义。

随着信息化技术的快速发展，互联网已经几乎走入了家家户户，移动互联网的迅速普及更是加快了信息网络的成长速度。群众参与公共事务讨论的网络平台渠道日益增多，在各类网络问政平台上均留下了大量文本信息。由于网络信息量巨大，依靠传统的分析技术和方法不仅给相关负责人带来巨额工作量，导致工作效率较低的同时也无法完全挖掘其中的价值。

互联网上来自用户的庞大的信息量使得政府更难全面地了解民情民意，政府管理者能否真正解决群众所关心的问题，能否及时给予相应的反馈，这与政府能否正确理解群众想表达的信息直接相关。而文本挖掘技术可以成为帮助管理者快速、高效、准确地定位海量文本信息的有效工具，进而可以帮助政府更加迅速地把握群众的思想动态和利益诉求，及时有效解决群众所关心的问题，最终加快服务型政府的构建与和谐社会的发展。

1.3 挖掘目标

本次建模利用网络问政平台的群众留言信息数据，采用各类数据挖掘技术，达到以下目标：

（1）建立关于群众留言内容的一级标签分类模型。

（2）定义热度评价指标，得出排名前五的热点问题及其对应的留言信息。

（3）设计针对相关部门对留言的答复意见质量的评价方案。

2.技术简介

2.1jieba 分词

“jieba”分词是一个 python 中文分词组件，可以对中文文本进行分词、词性标注、关键词抽取等功能，并且支持自定义词典。jieba 分词包整体的工作流程如下图所示：

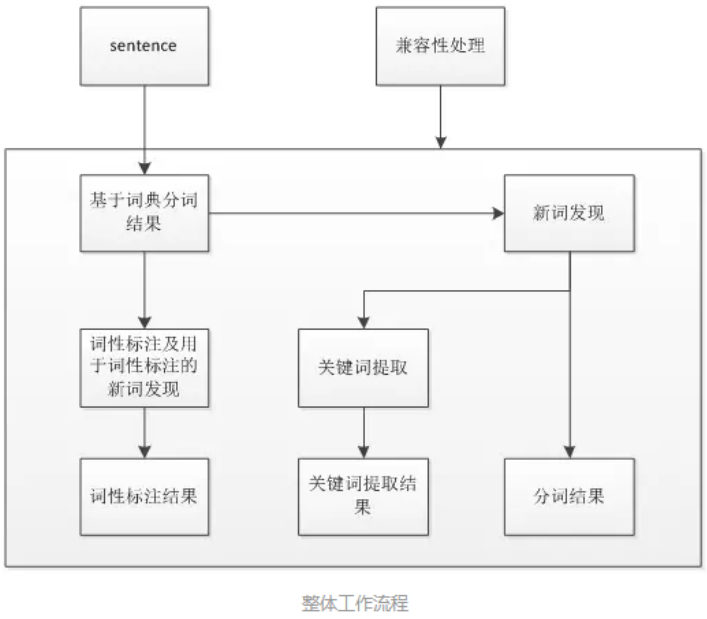

图 2.1 jieba 分词流程

2.1.1 分词

jieba 分词中， 首先通过对照词典生成句子的有向无环图，再根据选择的模式不同，根据词典寻找最短路径后对句子进行截取或直接对句子进行截取。对于未登录词（不在词典中的词）使用 HMM 进行新词发现。下图演示了分词的主要过程：

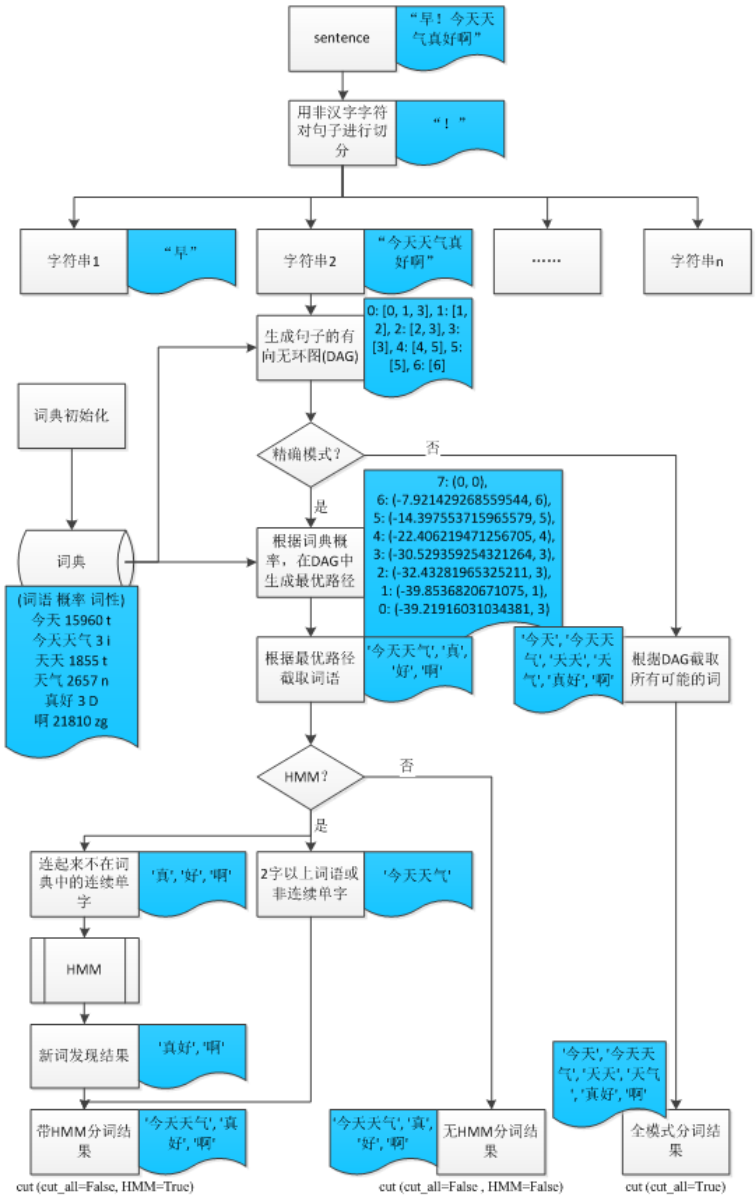

图 2.2 分词流程

2.1.2HMM 与新词发现

在 jieba 分词中，将字在词中的位置 B、E、M、S 作为隐藏状态，字是观测状态，使用了词典文件分别存储字之间的表现概率矩阵、初始概率向量和转移概率矩阵。这就是一个标准的解码问题，根据概率再利用 viterbi 算法对最大可能的隐藏状态进行求解。下图简单示范了 jieba 分词中新词发现模块的工作流程：

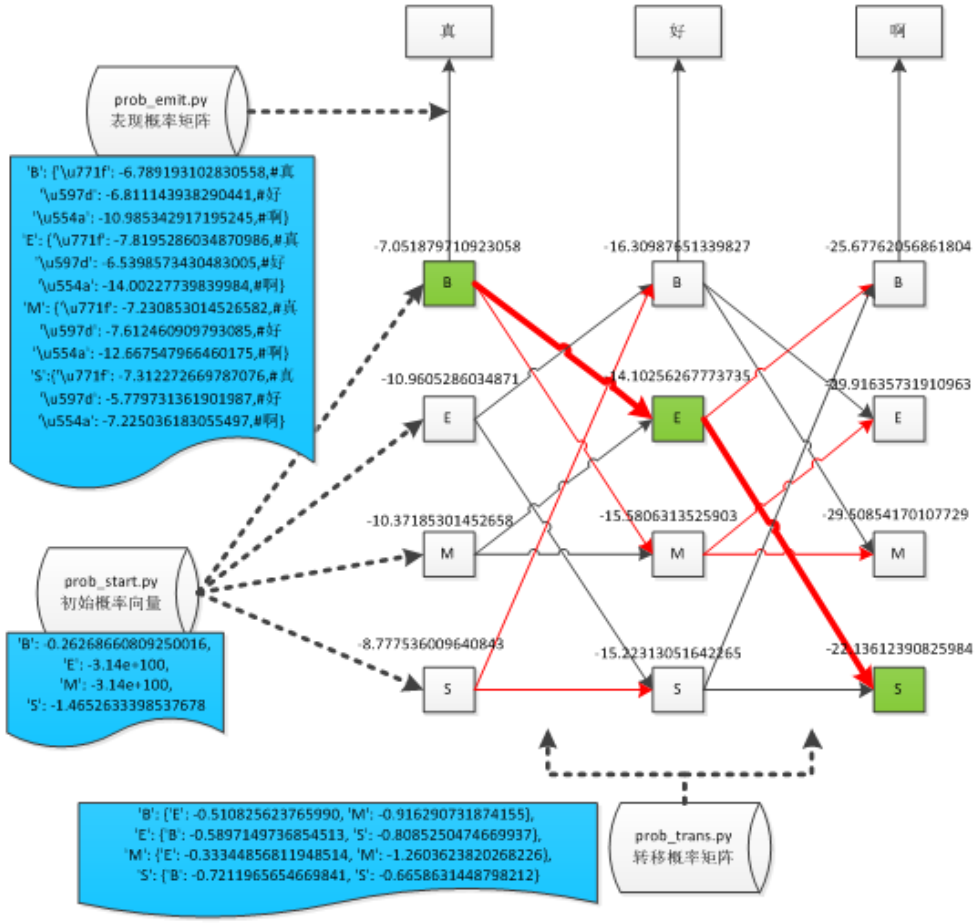

图 2.3 新词发现模块的工作流程

2.1.3 词性分析

词性分析部分与分词模块用了同一个基础的分词器，对于词典词的词性，将直接从词典中提取，但是对于新词，词性分析部分有一个专属的新词及其词性的发现模块。用于词性标注的 HMM 模型与用于分词的 HMM 模型相似，同样将文字序列视为可见状态，但是隐藏状态不再是单单的词的位置（B/E/M/S），而变成了词的位置与词性的组合，如（B,v）（B,n）（S,n）等等。具体的工作流程如下图所示：

图 2.4

2.1.4 关键词提取

jieba 分词中有两种不同的用于关键词抽取的算法，分别为 TextRank 和 TF-IDF。实现流程比较简单，其核心在于算法本身，下图为简单的实现流程：

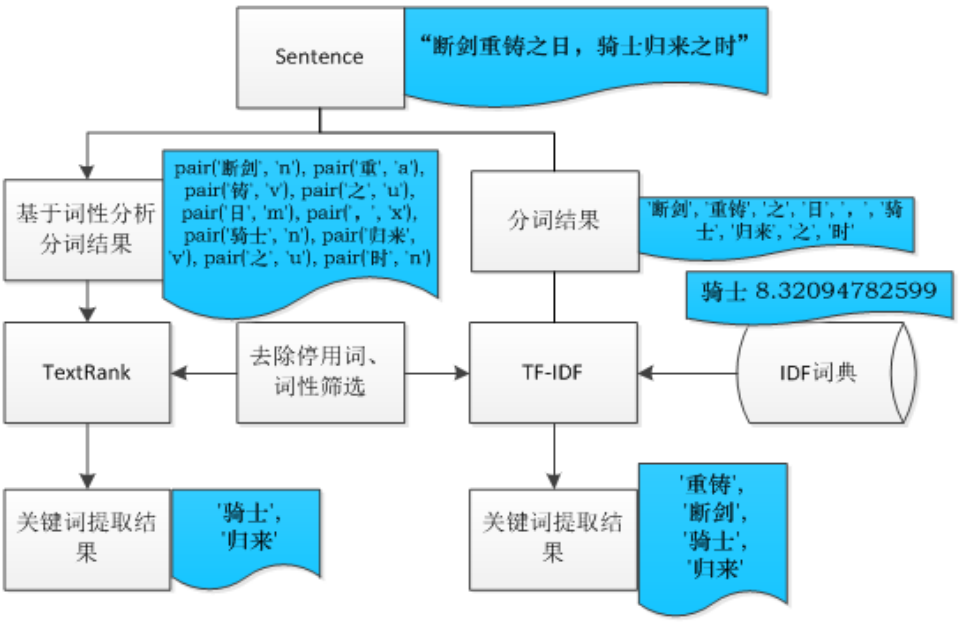

图 2.5

2.2TF-IDF 算法

TF-IDF 是一种用于信息检索与数据挖掘的常用加权技术。TF 是词频（Term Frequency），IDF 是逆文本频率指数（Inverse Document Frequency）。

TF-IDF 是一种统计方法，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 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。

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 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。如果某一类文档 C 中包含词条 t 的文档数为 m，而其它类包含 t 的文档总数为 k，显然所有包含 t 的文档数 n=m+k，当 m 大的时候，n 也大，按照 IDF 公式得到的 IDF 的值会小，就说明该词条 t 类别区分能力不强。但是实际上，如果一个词条在一个类的文档中频繁出现，则说明该词条能够很好代表这个类的文本的特征，这样的词条应该给它们赋予较高的权重，并选来作为该类文本的特征词以区别与其它类文档，这就是 IDF 的不足之处。

在一份给定的文件里，词频（TF）指的是某一个给定的词语在该文件中出现的频率。这个数字是对词数的归一化，以防止它偏向长的文件。对于在某一特定文件里的词语来说，它的重要性可表示为：$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$。上式中分子是该词在文件中的出现次数，而分母则是在文件中所有字词的出现次数之和。

逆向文件频率（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：$\mathrm{idf}_i = \lg \frac{|D|}{|\{j:t_i \in d_j\}|}$。其中，分子为语料库中的文件总数，分母为包含词语的文件数目，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$1 + |\{d \in D: t \in d\}|$作为分母。

然后再计算 TF 与 IDF 的乘积：$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \mathrm{idf}_i$。

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 常用于过滤常见的词语，保留重要的词语。

2.3LDA 建模

与一般的数据挖掘不同，文本挖掘的数据对象往往都是非结构化的，很难从信息中直接获取相关和期望的信息。尔主题模型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。

LDA 模型是一种生成式的主题模型，即狄利克雷分布（Latent Dirichlet Allocation，LDA）主题模型，其概率图模型如图所示，它将文档表示为主题的概率分布，而主题表示成词的概率分布，因此 LDA 可以被用来进行文本特征提取。

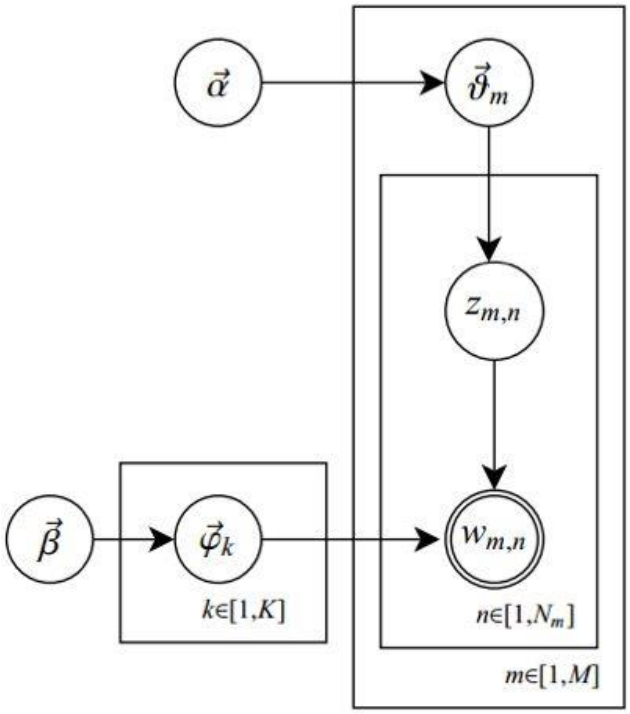

图 2.6 LDA 模型概念图

LDA 的模型解释

1)文档的主题先验分布服从参数为α 的 Dirichlet 分布，其中文档 d 的主题分布为$\theta_d = \text{Dirichlet}(\alpha)$

2)主题中的词的先验分布服从参数β 的先验分布，其中主题 k 的词分布为$\varphi_k = \text{Dirichlet}(\beta)$

3)文档 d 中的第 n 个词，从主题分布获得其主题编号分布为$z_{dn} = multi(\theta_d)$

4)文档 d 中的第 n 个词分布w_{dn} 的分布为$w_{dn} = multi(\varphi_{z_{dn}})$

其中 D 是训练数据集的大小，N 是一条训练数据的大小，K 是主题数。

LDA 的生成过程

LDA 假设文档是由多个主题的混合来产生的，每个文档的生成过程如下：

1. 从全局的泊松分布参数为β 的分布中生成一个文档的长度 N
2. 从全局的狄利克雷参数为 alpha 的分布中生成一个当前文档的θ
3. 对当前文档长度 NN 的每一个字都有

1) 从θ为参数的多项式分布生成一个主题的下标z_n

2) 从θ和 z 共同为参数的多项式分布中，产生一个字w_n

这些主题基于词的概率分布来产生词，给定文档数据集，LDA 可以学习出，是哪些主题产生了这些文档。

2.4 朴素贝叶斯算法

贝叶斯定理是关于随机事件 A 和 B 的条件概率（或边缘概率）的定理，贝叶斯定理公式如下:

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

其中，P(B|A)表示在事件 A 发生的前提下，发生事件 B 的概率；P(A|B)表示在事件 B 发生的前提下，发生事件 A 的概率；P(A)表示发生事件 A 的概率；P(B)表示发生事件 B 的概率。

朴素贝叶斯分类模型是贝叶斯分类法中最简单有效、实际应用较为成功的一种分类器，朴素贝叶斯分类法基于贝叶斯原理，其工作过程如下：

设 D 是训练集和其相关联的类标号集合。每个数据样本使用 n 维特征向量 X={x1，x2，…，xn}表示，描述了对 n 个特征样本 A1，A2，…，An 对元素组的 n 个度量。

若有 m 个类C_1、C_2,…，C_m，一个未知的数据样本（没有编号），分类器将会预测 X 属于具有最高后验概率（条件 X 下）的类。即朴素贝叶斯分类将未知的样本分配给C_i，其条件是$P(C_i\,|\,X\,)=P(C_j\,|\,X)$，$1\le j\le m$，$j\ne i$。最大化$P(C_i\,|\,X\,)$对应的类 Ci 概率大于其他类的概率。

根据大数定理（大数定理：设μ_n是 n 次独立试验中事件 A 发生的次数，且事件 A 在每次试验中发生的概率为 P，则对任意正数ε，有：$\lim_{n\to\infty}P\left(\left|\frac{\mu_n}{n}-p<\varepsilon\right|\right)=1$），当训练集包含充足的独立同分布样本时，$P\left(Y=C_j\right)$可通过各类样本出现的频率来进行估计，即当 n 足够大时，事件 A 出现的频率将几乎接近于其发生的概率，即频率的稳定性。

若类的先验概率未知，则假定这些类是等概率，即 $P(C_1)=P(C_2)=\ldots=P(C_m)$，因此需要将 $P(X|C_1)$ 转换为最大。类的先验概率可以用 $P(C_1)=|C_i,\ D|/|D|$，其中 $|C_i,\ D|$ 是 D 中 Ci 类的元组个数。具有很多属性的数据集，计算 $P(X|C_1)$ 的开销会变得很大，为降低计算开销，朴素贝叶斯分类法在估计类条件概率是假设属性间条件独立，即： $P(\mathrm{X}|C_i)\ldots P(x_1|C_i)$ $P(x_2|C_i)\ldots P(x_n|C_i)$ 为了预测 X 的类标号，对每个类 Ci，计算 $P(X|C_i)=P(C_i)$，预测元组 X 的类为 Ci，当且仅当 $P(X|C_i)P(C_i)>P(X|C_j)P(C_j)$，$1\le j\le m$，$j\ne m$，$j\ne i$，被预测的类标号就是使 $P(X|C_i)=P(C_i)$ 最大的 C_i，经过运算即可得到分类结果。

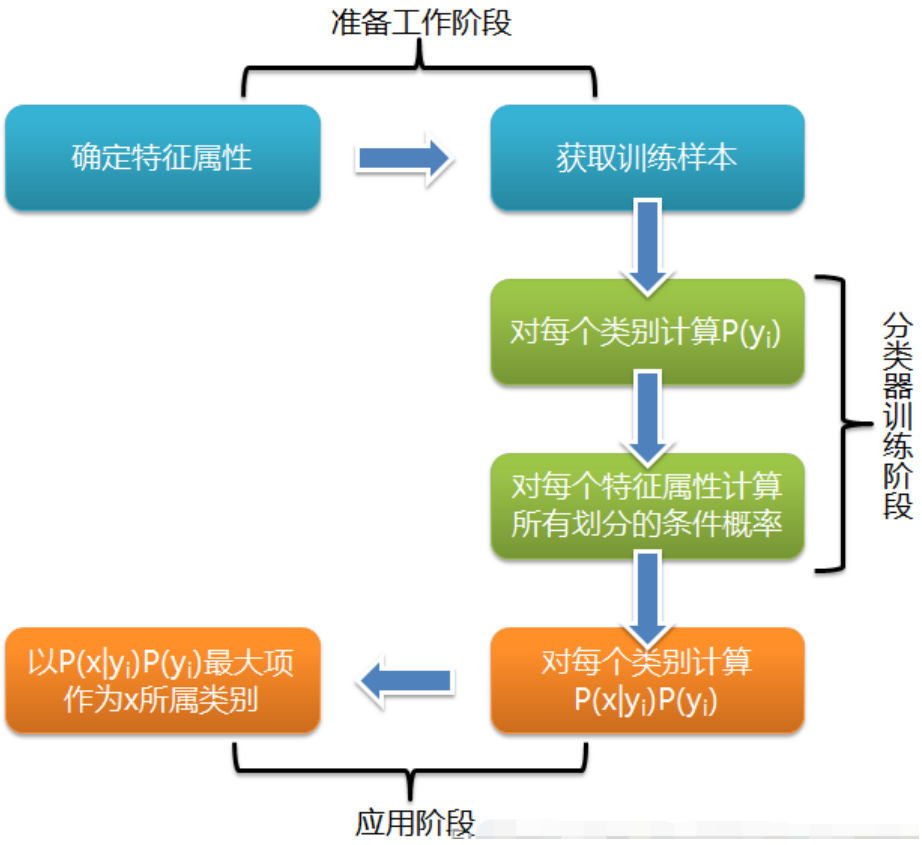

图 2.7 贝叶斯算法处理流程

2.5 LSI(潜语义分析)向量模型

LSI 是基于文档和词共现关系以及奇异值分解（SVD）方法来得到文本主题的一种模型，LSI 的基本原理是利用矩阵的奇异值分解，把高维空间中的词和文

档向量投影到更低维的空间使之降维并更易于得到两者之间的关系，将在高维空间中没有关系的文本（高维度文本向量不重合），在没有相同的词的情况下也可以用同样的向量进行表达。LSI 模型主要有 3 个步骤：

（1）模型构建与优化

LSI 模型首先需要用词-文档矩阵 A_{mn} 来表示整个文本库，词-文档矩阵是指由不同词在不同文本中的数量构成的矩阵。在 A_{mn} 中，m 表示文本库中不同词的数量，A_{mn} 中的每一行对应一个词；n 表示文本库中的文本总数量，A_{mn} 中的每一列则对应一个文本。则根据以上定义，矩阵中的某个元素 a_{ij} 就表示第 i 个词在第 j 个文本中出现的次数。但是因为 a_{ij} 的数量级可能会很大，也有可能为“0”，所以并不能采用 a_{ij} 的原始值，需要对其进行优化以后，再采用其值。对于 a_{ij} 优化值，本文使用局部权值计算函数 $L(i,j)$ 和全局权值计算函数 $C(i)$ 的乘积来表示

（2）降维

在第一步中，建立了高维度的 LSI 模型，但是因为该模型的维度太高，如果直接使用，会造成维度灾难，并不能产生我们想要的结果，因此在第二步中我们需要对第一步所建立的模型进行降维。首先根据公式对矩阵 A_{mn} 进行了奇异值分解：

$$A = U\sum V^T$$

式（3）中，V 与 U 分别表示 A_{mn} 的奇异值对应的左、右奇异向量矩阵，并有 $V^TV = U^TU = I$；Σ 是把 A_{mn} 中的奇异值依照递减排列构成的对角矩阵，并取 U 和 V 的前 d 列（即将矩阵的维度降低到 d 维）$\left(d \ll min(m,n)\right)$，得到公式：

$$A_d = U_d \sum_d V_d^T$$

降维之后，A_{mn} 的近似值用 A_d 表示。U_d 和 V_d 的行向量分别表示词向量和文档向量，所以 d 值的选择将对检索效率和质量产生很大的影响，d 值过大会导致词与词之间本来所具有的关联性无法得到表示，不能达到语义检索的目的；d 值

过小会导致丢失可靠信息。

（3）文本相似度计算

通过 LSI 模型得到文本主题矩阵之后使用余弦相似度的方法就可以用该矩阵进行文本相似度的计算。如公式所示:

$$SIM(d_1, d_i, ...) = \frac{\sum_{i=2}^{n} d_1 \times d_i}{\prod_{i=2}^{n} \sqrt{(d_1)^2}\sqrt{(d_i)^2}}$$

2.6 Python 自带库

2.6.1 gensim

gensim 是一个 python 库，可以用来做一些简单的文本处理——文本相似度比较。

简单来讲，gensim 的主要功能有：把文本转为向量，抽取文本中的关键词，比较两个文本的相似度，或是计算一个查询（本质也是一个文本）与一个文档集合中所有文档的关联程度。它有三个主要模块：corpora，models 和 similarites，分别提供不一样的功能。

用 gensim 计算文本相似度的方法就是先用 corpora 模块将文档转化为简单的稀疏矩阵，然后用 models 模块得到符合需要的向量模型，最后用 similarites 模块计算相似度。

2.6.2 CountVectorizer

CountVectorizer()是一种文本特征提取函数，它只考虑每个单词出现的频率，然后构成一个特征矩阵，每一行表示一个训练文本的词频统计结果。其思想是：先根据所有训练文本，不考虑其出现顺序，只将训练文本中每个出现过的词汇单独视为一列特征，构成一个词汇表（vocabulary list），该方法又称为词袋法（Bag of Words）。

CountVectorizer 是通过 fit_transform 函数将文本中的词语转换为词频矩阵，矩阵元素a_{ij}表示 j 词在第 i 个文本下的词频。即各个词语出现的次数，通过

get_feature_names()可看到所有文本的关键字，通过 toarray()可看到词频矩阵的结果。

2.6.3 TfidfTransformer

TfidfTransformer()是一种权重计算函数，基于 TF-IDF 算法。此算法包括 TF 算法与 IDF 算法，两者相乘得到 TF-IDF 算法。

2.6.4 MultinomialNB 类

MultinomialNB 假设特征的先验概率为多项式分布，即如下式：

$$P(X_j = x_{jl} | Y = C_k) = \frac{x_{jl} + \lambda}{m_k + n\lambda}$$

其中，$P(X_j = x_{jl} | Y = C_k)$是第 k 个类别的第 j 维特征的第 1 个个取值条件概率。m_k是训练集中输出为第 k 类的样本个数。λ 为一个大于 0 的常数，常常取为 1，即拉普拉斯平滑。也可以取其他值。

MultinomialNB 参数有 3 个，其中，参数 alpha 即为上面的常数 λ，如果你没有特别的需要，用默认的 1 即可。如果发现拟合的不好，需要调优时，可以选择稍大于 1 或者稍小于 1 的数。布尔参数 fit_prior 表示是否要考虑先验概率，如果是 false,则所有的样本类别输出都有相同的类别先验概率。否则可以自己用第三个参数 class_prior 输入先验概率，或者不输入第三个参数 class_prior 让 MultinomialNB 自己从训练集样本来计算先验概率，此时的先验概率为$P(Y = Ck) = \frac{m_k}{m}$。其中 m 为训练集样本总数量，$m_k$为输出为第 k 类别的训练集样本数。总结如下：

fit_prior	class_prior	最终先验概率
false	填或者不填均可	$P(Y = C_k) = 1/k$
true	不填	$P(Y = Ck) = \frac{mk}{m}$
true	填	$P(Y = Ck) = class\_prior$

3.具体思路与流程

3.1 建立关于群众留言内容的一级标签分类模型

3.1.1 数据预处理

3.1.1.1 观察数据，数据预处理

题目给出的数据中，给出了留言编号、留言用户、留言主题、留言时间、留言详情和一级标签 6 项数据，根据问题一的需要我们去除了留言用户和留言时间两项数据，将处理后的数据保存为 data2.csv

将 data2.csv 导入 Python，给四列数据分别赋值为'number'、'theme'、'content'、'label'，将留言详情的文本内容转换为 list 格式，并得到数据的大小 9210 条数据。

```
#1.2转换为list格式
print(df_news.shape)
content = df_news.content.values.tolist()
print (content[1000])
```

```
(9210, 4)
```

K2区K1区北路谢氏足部保健中心那栋楼(紫金丰泰斜对面)后面乱搭乱建了一排木屋棚子,影响居民过路和美观,如果大家都这样在后面乱搭建还怎么过路,那成什么样子,要求拆除违章乱建的木屋棚子,不要搞到大家都学样都这样乱搭~

图 数据预处理结果

3.1.1.2 对留言详情进行中文分词

在对留言详情信息进行挖掘分析之前，先要把非结构化的文本信息转换为计算机能够识别的结构化信息。为了便于转换，先要对这些中文文本信息进行中文分词。我们采用 python 的中文分词包 jieba 进行分词。

jieba 采用了基于前缀词典实现的高效词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图 (DAG)，同时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，使得能更好的实现中文分词效果。

```
#1.3使用jieba分词
content_S = []
for line in content:
    current_segment = jieba.lcut(line)
    if len(current_segment) > 1 and current_segment != '\r\n': #换行符
        content_S.append(current_segment)
print(content_S[1000]) #(list of list)
```

图 3.1 jieba 分词程序

```
['\n', '\t', '\t', '\t', '\t', '\t', '\n', '\t', '\t', '\t', '\t', '\t', '\u
3000', '\u3000', 'K2', '区', 'K1', '区', '北路', '谢氏', '足部', '保健', '中
心', '那栋', '楼', '(', '紫金', '丰泰', '斜对面', ')', '后面', '乱', '搭乱
建', '了', '一排', '木屋', '棚子', ',', '影响', '居民', '过路', '和', '美
观', ',', '如果', '大家', '都', '这样', '在', '后面', '乱', '搭建', '还',
'怎么', '过路', ',', '那成', '什么', '样子', ',', '要求', '拆除', '违章',
'乱建', '的', '木屋', '棚子', ',', '不要', '搞', '到', '大家', '都', '学样',
'都', '这样', '乱', '搭', '~', ' ', '\n', '\t', '\t', '\t', '\t', '\t',
'\n', '\t', '\t', '\t', '\t', '\t', '\n', '\t', '\t', '\t', '\t']
```

图 3.2 部分分词结果示意图

3.1.1.3 生成词云图

为节省存储空间和提高搜索效率，在生成词云图之前我们先进行停用词过滤。经过 jieba 分词的文本数据存在某些表达无意义的字或词，这些字或词即被称为 Stop Words（停用词）。停用词有两个特征：一是极其普遍、出现频率高；二是包含信息量低，对文本标识无意义。

为了找出这些停用词，我们导入停用词表 stopword.txt，去除停用词。再使用 matplotlib 与 wordcloud 模块对过滤停用词后的文本生成词云。

```
#1.5使用停用词
stopwords=pd.read_csv("stopword.txt",index_col=False,sep="haha",quoting=3,names=['stopwor
d']) #list
#print(stopwords.head(20))


def drop_stopwords(contents, stopwords):
    contents_clean = []
    all_words = []
    for line in contents:
        line_clean = []
        for word in line:
            if word in stopwords:
                continue
            line_clean.append(word)
            all_words.append(str(word))
        contents_clean.append(line_clean)
    return contents_clean, all_words
    # print (contents_clean)
contents = df_content.content_S.values.tolist()
stopwords = stopwords.stopword.values.tolist()
contents_clean, all_words = drop_stopwords(contents, stopwords) ###使用停用词
df_content=pd.DataFrame({'contents_clean':contents_clean}) ##每一列的分词
print(df_content.head())
df_all_words=pd.DataFrame({'all_words':all_words})  ###所有词语
print(df_all_words.head())
```

图 3.3 去停用词

```
#1.6查看词频
import numpy
words_count=df_all_words.groupby(by=['all_words'])['all_words'].agg({"count":numpy.size})
##分组大小
print(words_count.head())
words_count=words_count.reset_index()
print(words_count.head())
words_count=words_count.sort_values(by=["count"],ascending=False)##排序
print(words_count.head())
```

```
#1.7生成词云
%matplotlib inline
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['figure.figsize'] = (10.0, 5.0)
wordcloud=WordCloud(font_path="./data/simhei.ttf",background_color="white",max_font_size=
80)
word_frequence = {x[0]:x[1] for x in words_count.head(100).values}
wordcloud=wordcloud.fit_words(word_frequence)
plt.imshow(wordcloud)
plt.show()
```

图 3.4 生成词云

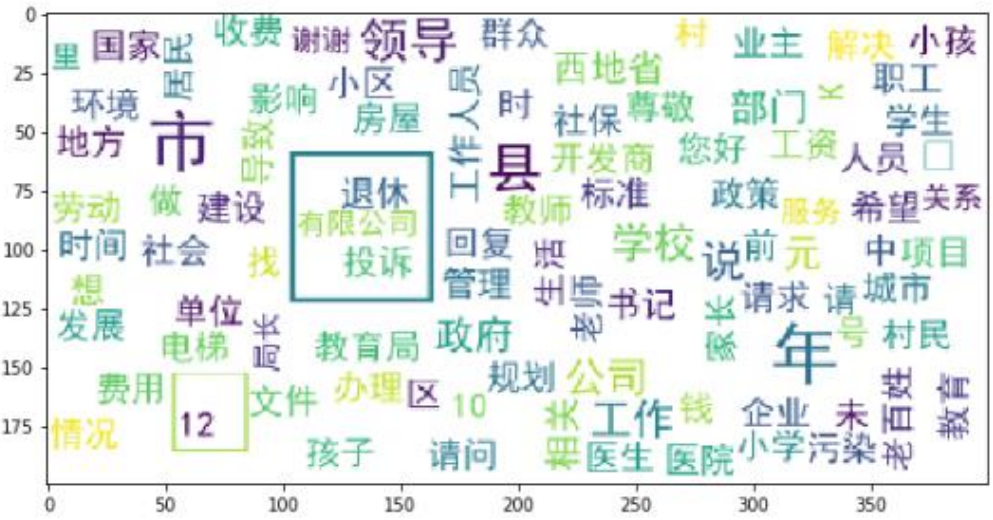

图 3.5 留言详情信息词云图

3.1.2.TF-IDF 关键词提取

```
##2.TF-IDF关键词提取
#2.1 提取关键词
import jieba.analyse
index = 2400
print (df_news['content'][index])
content_S_str = "".join(content_S[index])
print ("  ".join(jieba.analyse.extract_tags(content_S_str, topK=5, withWeight=False)))#3
输出前五个关键词
```

图 3.6 TF-IDF 算法

得到前五个关键词为环评、厂子、生产、环保部门、易拉罐

3.1.3LDA 建模

利用 LDA 这种矩阵分解技术，将留言信息表示为文档矩阵，如下表的矩阵

	W_1	W_2	…	W_m
D_1	0	2	…	3
D_2	1	4	…	0
…	…	…	…	…
D_n	1	1	…	0

表 3.1 语料库概念模型

其中，N 个文档 D_1， D_2，…D_n 的组成语料库，M 个词 W_1， W_2，…W_n 组成

词汇表。矩阵中的值表示了词W_j在文档D_i中出现的频率，同时，LDA 将这个矩阵转换为两个低维度的矩阵，M_1和M_2。

	Z_1	Z_2	…	Z_k
θ_1	0	2	…	3
θ_2	1	4	…	0
…	…	…	…	…
θ_n	1	1	…	0

表 3.2 M_1矩阵

M_1矩阵是一个 N×K 维的 document-topic 矩阵，N 指文档的数量，K 指主题的数量，M_1中，θ_i是一个长度为 k 的向量，用于描述当前文档θ_i在 k 个主题上的分布情况，Z 表示具体的主题。

	W_1	W_2	…	W_m
ϕ_1	0	2	…	3
ϕ_2	1	4	…	0
…	…	…	…	…
ϕ_k	1	1	…	0

表 3.3 M_2矩阵

M_2矩阵是一个 K×V 维的 document-topic 矩阵， K 指主题的数量，V 指词汇表的大小。M_2中每一行都是一个ϕ 分布，也就是主题ϕ_k在 m 个词上的多项式分布情况。

```
##3.LDA建模
#3.1建模
import gensim
from gensim import corpora
from gensim import models
from gensim import similarities
```

```
#做映射，相当于词袋
dictionary = corpora.Dictionary(contents_clean) ##格式要求：list of list形式，分词好的的整个语料
corpus = [dictionary.doc2bow(sentence) for sentence in contents_clean]  #语料
lda = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=dictionary, num_topics=20) #类似Kmeans自己指定K值
print (lda.print_topic(1, topn=5)) ##第一个主题，关键词5个
```

```
0.026*"公司" + 0.020*"年" + 0.012*"职工" + 0.012*"  " + 0.012*" "
```

图 3.7

经过迭代，我们得到 20 个主题的关键词和权重：

```
#3.2 输出20个主题的的关键词和权重
for topic in lda.print_topics(num_topics=20, num_words=5):
    print (topic[1])
```

```
0.001*" " + 0.001*"市" + 0.001*"  " + 0.000*"年" + 0.000*"
"
0.026*"公司" + 0.020*"年" + 0.012*"职工" + 0.012*"  " + 0.012*" "
0.016*" " + 0.011*"网吧" + 0.006*"贵委" + 0.006*"商品房" + 0.004*"鸡鸭"
0.066*"业主" + 0.035*"物业" + 0.030*"开发商" + 0.017*"电梯" + 0.016*"   "
0.308*"  " + 0.064*"  " + 0.020*"生育" + 0.014*"
" + 0.014*"年"
0.002*" " + 0.001*"市" + 0.001*"  " + 0.001*"房屋" + 0.001*"业主"
0.033*"  " + 0.024*"幼儿园" + 0.007*"配套" + 0.007*"恒" + 0.006*"小区"
0.011*" " + 0.007*"校车" + 0.006*"培训班" + 0.003*"  " + 0.003*"客车"
0.032*" " + 0.030*"旅游" + 0.020*"  " + 0.017*"景区" + 0.015*"游客"
0.004*" " + 0.004*"票" + 0.004*"追查" + 0.004*"  " + 0.003*"伪证"
0.425*" " + 0.048*"
" + 0.044*"  " + 0.010*"市" + 0.009*"年"
0.045*"  " + 0.032*"市" + 0.023*"  " + 0.018*"计生" + 0.016*"产品"
0.179*" " + 0.064*"
" + 0.020*"  " + 0.007*"市" + 0.006*"说"
0.030*"  " + 0.015*"年" + 0.014*"患者" + 0.012*"
" + 0.011*"考试"
0.003*"  " + 0.002*"    " + 0.001*"  " + 0.001*"业主" + 0.001*"
"
0.044*"学生" + 0.043*"学校" + 0.038*"孩子" + 0.036*"老师" + 0.022*"家长"
0.023*"A9" + 0.007*"    " + 0.006*"市" + 0.005*"发展" + 0.002*"
"
0.011*" " + 0.009*"D5" + 0.009*"订立" + 0.006*"  " + 0.003*"燃气"
0.045*"  " + 0.025*"
" + 0.018*"     " + 0.016*"医院" + 0.011*"年"
0.024*"患者" + 0.020*"报销" + 0.016*"失业" + 0.015*"
" + 0.014*"     "
```

图 3.8

3.1.4 基于贝叶斯算法进行分类

```
###4.基于贝叶斯算法进行新闻分类
#4.1 转换数据
df_train=pd.DataFrame({'contents_clean':contents_clean,'label':df_news['label']})
print(df_train.tail())
print(df_train.label.unique())
```

```
                                          contents_clean label
9206  [\n,  ,  ,  ,  ,  , \n,  ,  ,  ,  ,  ,  ,  , 夫...  卫生计生
9207  [\n,  ,  ,  ,  ,  , \n,  ,  ,  ,  ,  ,  ,  , 2...  卫生计生
9208  [\n,  ,  ,  ,  ,  , \n,  ,  ,  ,  ,  ,  ,  , 再...  卫生计生
9209  [\n,  ,  ,  ,  ,  , \n,  ,  ,  ,  ,  , K8, 县惊现...  卫生计生
9210  [\n,  ,  ,  ,  ,  , \n,  ,  ,  ,  ,  ,  ,  , 领...  卫生计生
['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']
```

```
label_mapping = {"一级标签": 1, "城乡建设": 2, "环境保护": 3, "交通运输": 4, "教育文体":
5, "劳动和社会保障": 6,"商贸旅游": 7,"卫生计生": 8}
df_train['label_new'] = df_train['label'].map(label_mapping) ##变换label
print(df_train.head())
```

图 3.9

3.1.4.1 创建训练、测试集

```
#4.2 创建训练、测试集
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df_train['contents_clean'].values, df
_train['label'].values, random_state=1)
#print(x_train)
#print(x_train[0])
#x_train = x_train.flatten()
words = []
for line_index in range(len(x_train)):
    try:
        #x_train[line_index][word_index] = str(x_train[line_index][word_index])
        words.append(' '.join(x_train[line_index]))
    except:
        print (line_index)
print(words[0])
print (len(words))
```

图 3.10

3.1.4.2 建立贝叶斯模型和 TF-IDF 模型

```
#4.3 构建贝叶斯模型
from sklearn.feature_extraction.text import CountVectorizer
vec = CountVectorizer(analyzer='word', max_features=4000,  lowercase = False)
vec.fit(words)
```

```
#4.4 TF-IDF模型
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(analyzer='word', max_features=4000,  lowercase = False)
vectorizer.fit(words)
```

图 3.11 分别建立模型

由于本次样本特征大部分是多元离散值，所以选择对两个模型均使用先验为

多项式分布的朴素贝叶斯 MultinomialNB 类进行验证

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(vec.transform(words), y_train)
test_words = []
for line_index in range(len(x_test)):
    try:
        #x_train[line_index][word_index] = str(x_train[line_index][word_index])
        test_words.append(' '.join(x_test[line_index]))
    except:
         print (line_index)
print('test_words[0]',test_words[0])
print('test_words_sorce',classifier.score(vec.transform(test_words), y_test))
```

图 3.12 贝叶斯模型验证

```
#4.4 TF-IDF模型
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(analyzer='word', max_features=4000,  lowercase = False)
vectorizer.fit(words)

from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(vectorizer.transform(words), y_train)
print(classifier.score(vectorizer.transform(test_words), y_test))
```

图 3.13 TF-IDF 模型验证

分别得到分数为 0.865392965696917 和 0.8610508033000435

3.2 定义热度评价指标，得出排名前五的热点问题

3.2.1 数据预处理

3.2.1.1 转换数据类型

为便于后续分析，提取附件 3 的留言主题和留言信息并保存为文本形式，这是为了将两种留言信息以完整的文章形式分析，结果分别保存在 data3_title.csv 及 data3_train.csv，详细程序见附件 data3_train.py

3.2.1.2 中文分词，去停用词，词性筛选

由于中文文本的特点是词与词之间没有明显的界限，从文本中提取词语时需要分词，我们编写了 dataPrepos 函数对留言主题和留言详情的文本进行中文分词，同时可以达到去停用词和词性筛选的效果。

其中，中文分词我们选择采用 Python 开发的一个中文分词模块——jieba 分词，jieba 分词用到的算法：

```python
# 数据预处理操作：分词，去停用词，词性筛选
def dataPrepos(text,stopkey):
    l = []
    pos = ['n','nz','v', 'vd', 'vn', 'l', 'a', 'd']  # 定义选取的词性
    seg = jieba.posseg.cut(text) #分词
    for i in seg:
        if i.word not in stopkey and i.flag in pos:# 去停用词 + 词性筛选
            l.append(i.word)
    return l
```

图 3.14 dataPrepos 函数

3.2.2 获取文本关键词

编写 getKeywords_tfidf 函数获取文本的 top10 关键词，函数编写思路如下，首先将转换类型后的文档输入一个 list 中，并调用编写好的 dataPrepos 函数对文本进行预处理，连接成若干词语组成的字符串。利用 Python 的 sklearn 模块下的文本特征提取函数 CountVectorizer，将文本中的词语转换为词频矩阵，再利用同在 sklearn 模块下的权重计算函数 TfidfTransformer 类统计每个词的 TF-IDF 权值，获取词袋模型中的关键词，得到该文本的 TF-IDF 矩阵，最后打印所有词语权重，详细程序见附件 dataTFIDF.py

图是部分结果显示：

```
————这里输出第 1 篇文本的词语tf-idf————
一业 0.002308410211967499
一事 0.009233640847869996
一体化 0.002308410211967499
一卡通 0.002308410211967499
一号线 0.002308410211967499
一小 0.002308410211967499
一师 0.004616820423934998
一期 0.018467281695739992
一村 0.002308410211967499
一楼 0.004616820423934998
一直 0.03462615317951248
```

图 3.15 附件 3 留言主题的词语 TF-IDF 值

我们将输出的留言主题和留言详情的词语 TF-IDF 值结果分类汇总保存在 data3TFIDF.xlsx 中，

留言详情	小区	医院	没有	当事人	政府	业主	解决	已经	房屋	孩子
留言主题	问题	小区	扰民	严重	街道	反映	咨询	投诉	建议	国际

表 3.4 留言详情和留言主题的 top10 关键词

3.2.3.文本相似度比较

相似度比较我们选择使用Python的gensim模块计算关键词字符串与附件三的中重点问题的相似度。利用 gensim 计算文本相似度，首先用 copora 模块把文档转为简单的稀疏矩阵；然后用 models 模块得到符合需要的向量模型；最后用 similarities 模块计算相似度。

3.2.3.1 挑选热点问题

热点问题，即某一时段内群众集中反映的某一问题；群众的集中反映有两个维度的体现，一是有多个用户反映该问题，我们通过文本相似度来量化这一维度；另一方面，点赞数也可以反映群众对该问题的关心程度，因此我们对附件三中的留言点赞数进行排序，选取了点赞数排名前 20，即点赞数在 40 以上的 20 个问题作为备选的重点问题，从中挖掘出 top5 的热点问题，并对其进行编号

问题 ID	留言主题
1	’A 市 A5 区汇金路五矿万境 K9 县存在一系列问题’,
2	’反映 A 市金毛湾配套入学的问题’,
3	’请书记关注 A 市 A4 区 58 车贷案’,
4	’严惩 A 市 58 车贷特大集资诈骗案保护伞’,
5	’承办 A 市 58 车贷案警官应跟进关注留言’,
6	’A4 区绿地海外滩小区距长赣高铁最近只有 30 米不到，合理吗？’,
7	’A 市富绿物业丽发新城强行断业主家水’,
8	’建议西地省尽快外迁京港澳高速城区段至远郊’
9	’请问 A 市为什么要把和包支付作为任务而不让市场正当竞争？’
10	’关于 A6 区月亮岛路沿线架设 110kv 高压线杆的投诉’
11	’A 市三一大道全线快速化改造何时启动？’
12	’关于 A6 区月亮岛路 110kv 高压线的建议’
13	’A 市长房云时代多栋房子现裂缝，质量堪忧’
14	’建议 A 市经开区收回东六路恒天九五工厂地块，打造商业综合体’
15	’反映 A 市地铁 3 号线松雅湖站点附近地下通道问题’
16	’建议加大 A7 县东六线榔梨段拆迁力度’
17	’A3 区郝家坪小学什么时候能改扩建?’

18	’问问 A 市经开区东六线以西泉塘昌和商业中心以南的有关规划’
19	’建议 A7 县尽快出让星沙滨湖路以南，特立路以北的土地’
20	’关于加快修建 A 市南横线的建议’

表 3.5 点赞数 top20 重点问题

3.2.3.2 文本转向量

文本转向量算法的原理是统计文本中每个单词出现的个数，所以我们还要把由 20 个重点问题组成文档集里的每个文档切词，也就是说把文档编程一个单词列表。这里我们再次采用了 Python 的 jieba 分词模块进行分词。将分词结果构建文档集（经过分词后转换为单词列表集）的词典，再利用词典来用向量表示文档，最后应用 models 模块将这个较为简单的向量模型转换为 LSI(潜语义分析)向量模型来做相似度计算。

3.2.3.3 相似度计算

首先将 20 个关键词变成查询输入模型中，利用 similarities.MatrixSimilarity 类建立索引，得到相似度，最终整理得到关键词-重点问题相似度矩阵

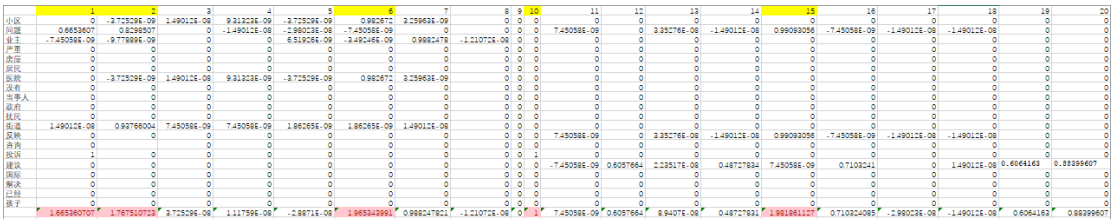

图 3.16 关键词-重点问题相似度矩阵

由图可知，其中文本相似度最高的为 15 号、6 号、2 号、1 号及 10 号问题。

热度排名	问题 ID	热度指数	时间范围	地点/人群	问题描述
1	15	1.981861	2019/08/26 至 2019/09/06	A 市地铁 3 号线松雅西地省站	A 市地铁 3 号线松雅西地省站地下通道建设问题
2	6	1.96	2019/04/17	A4 区绿地海外	长赣高铁征地路线对 A4 区

		5344	至	滩小区	绿地海外滩小区影响巨大
			2019/09/07		
3	2	1.76	2019/03/12	梅溪湖中学周	梅溪湖中学周边业主子女
		7511	至	边业主	入学问题
			2019/04/11		
4	1	1.66	2019/05/05	A 市五矿万境	A 市 A5 区汇金路五矿万境
		5361	至	K9 县	K9 县存在一系列问题
			2019/11/11		
5	10	1	2019/03/26	A6 区月亮岛路	关于 A6 区月亮岛路沿线架
			至		设 110KV 高压电线杆的投诉
			2019/04/15		

表 3.6 热点问题表

3.3 设计针对相关部门对留言的答复意见质量的评价方案

3.3.1 答复意见质量的可靠性

可靠性是指平台能够准确可靠地提供所承诺服务的能力。与传统的服务相比，智慧政务平台服务的内容以信息的形式呈现，信息的可靠性、质量的高低、更新的及时性是用户关注的焦点。可靠性维度是指平台能够准确、及时地向用户提供服务，并且可以保证服务质量和履行承诺。具体体现在信息的可靠性和办事能力的可靠性两方面。提供的信息是否真实、全面、能否做到及时更新，各项服务功能是否均可使用，是否具备承诺服务的能力。可靠性反映的是政府的公信力，只有可靠性维度达标，公众才会使用这个平台。

3.3.2 答复意见质量的响应性

响应性是指答复意见平台能否及时迅速地响应公众对服务的需求。具体体现在网页链接的速度、检索效果、处理正常政务和投诉事务的响应速度，提供服务的在线工作人员能否在承诺的时间内答复问题。

留言的平均回复时长是衡量政府回应速度的重要指标。根据各地方网络问政

管理办法，如《阳江市网络问政管理暂行办法》规定，对收到的各类信件要及时进行回复，一般信件在网民来信后(批转件以各地各部门收件日期计算)5 个工作日内向网民作出处理意见。各地各部门遇到 5 个工作日内无法办结的复杂问题可申请延时，但须在 5 个工作日内作初步回复。

我们通过对附件 4 利用 DATADIF 函数进行整理得出回应时长，分类汇总以及绘图得出留言回应时长的折线图（图一）。整体上，民众诉求的平均回应天数超过 20 天，民众诉求的回应天数集中在 0-28 天时段内，但任然有部分的留言时长较长，甚至长达 1161 天。这意味着政府科层体系对民众诉求的回应效率一般。对于复杂留言的处理时长较长，且没有做到“各地各部门遇到 5 个工作日内无法办结的复杂问题可申请延时，但须在 5 个工作日内作初步回复。”的要求，政府需要加强政务系统信息管理，争取初步实现“5 天”答复。

	回应时长		回应时长		回应时长		回应时长
0 计数	59	43 计数	14	88 计数	2	186 计数	1
1 计数	172	44 计数	11	89 计数	3	192 计数	1
2 计数	136	45 计数	7	90 计数	2	195 计数	4
3 计数	148	46 计数	3	91 计数	2	198 计数	1
4 计数	149	47 计数	3	93 计数	2	202 计数	1
5 计数	132	48 计数	6	94 计数	2	210 计数	2
6 计数	130	49 计数	5	95 计数	1	215 计数	1
7 计数	124	50 计数	6	96 计数	1	223 计数	1
8 计数	121	51 计数	7	98 计数	1	240 计数	1
9 计数	88	52 计数	5	99 计数	2	244 计数	2
10 计数	82	53 计数	7	100 计数	1	252 计数	1
11 计数	77	54 计数	5	101 计数	5	255 计数	1
12 计数	83	55 计数	9	102 计数	1	279 计数	1
13 计数	99	56 计数	2	104 计数	1	297 计数	2
14 计数	84	57 计数	3	105 计数	1	301 计数	1

15 计数	71	58 计数	5	107 计数	1	343 计数	1
16 计数	62	59 计数	5	109 计数	1	408 计数	1
17 计数	62	60 计数	9	110 计数	3	894 计数	1
18 计数	47	61 计数	2	111 计数	2	1161 计数	1
19 计数	59	62 计数	1	112 计数	2	**总计数**	**2816**
20 计数	37	63 计数	6	114 计数	2		

表 3.7

图回应时长计数表

与传统服务可以面对面的交流不同，公众通过平台这个媒介提供服务，因此特别注重平台的响应性。与电子政务相比，智慧政务具有主动服务、快速反应的特征，响应性维度的指标构建也体现了智慧政务“回应的政府”的政务服务理念。

3.3.3 答复意见质量的可解释性

广义上的可解释性指在我们需要了解或解决一件事情的时候，我们可以获得我们所需要的足够的可以理解的信息。政府对民众诉求的解释指的是对相关政策、事情、业务作解释与说明，这体现了对民众诉求的深层次回应，良好的解释能够改善民众与政府之间的关系，增进彼此间的信任。从政府对民众诉求的解释说明

程度来看，回复内容的字符平均长度为 360.60，其中答复意见字符数最长的是留言编号 96762 的长达 7883 的一段答复意见，详细具体的回答了关于市民的扶贫工作疑惑，这说明政府对于民众的诉求是切实具体的有效回应了。

J	K	L	M	N
	回应时长	回应时长平均值	答复意见字符数	答复意见字符数平均值
	0	20.31252109	105	360.600142
	0		382	
	0		1433	答复意见字符数最大值
	0		200	7883

图 3.17 答复意见统计

4.总结

总结本次比赛，我们基于 jieba 分词法，对留言信息进行中文分词，构建词云图，进一步运用 TF-IDF 对关键词进行提取，再利用 LDA 技术将留言信息表示为文档矩阵，从而得到 20 个主题的关键词和权重，进而使用贝叶斯算法进行分类；利用 TF-IDF 得到文本中词组的权重进行排序后，运用 gensim 进行文本相似度比较，得到了排名前五的热点问题；从留言答复意见的可靠性、响应性、可解释性方面对留言答复意见的质量进行分析，设计了针对留言答复意见质量的评价方案。

但是我们在最后一问构建的评价方案没有做到完全实现，一方面是我们的评价方案较为复杂，另一方面也是当前文本挖掘模型的不足，我们将在后期进行深入的研究。

5. 参考文献

[1]李少温.基于网络问政平台大数据挖掘的公众参与和政府回应问题的研究。

[2]. 郑宝，基于政府治理能力现代化的智慧政务建设探析. 改革与开放，2019(8): 第 25-27 页.

[3]. 邵梓捷._钟摆式回应_回应性不足的一种解_省略_基于 S 市地方领导留言板的实证研究

[4]. 王颖，基于 SERVQUAL 模型的智慧政务服务质量评价研究，2019，华中科技大学. 第 78 页.

[5]. 周洲等, 基于 TF-IDF 及 LSI 模型的主观题自动评分系统研究. 软件, 2019. 40(02): 第 158-163 页.

[6]. 杨涛, 基于标签相关性的文本多标签分类算法的研究, 2019, 北京工业大学. 第 78 页.

[7]. 杨涛, 基于标签相关性的文本多标签分类算法的研究, 2019, 北京工业大学. 第 78 页.

[8]. 陈欢,.结合LDA与Self_Attention的短文本情感分类方法