“智慧政务”中的文本挖掘作用

摘　要

智慧政务系统凭借其办公效率高、通信成本低、管理简便等优点，现已成为政府对民众进行智能监管、智能决策以及提供智能服务的重要渠道。因此，对“智慧政务”中的文本进行挖掘分析，了解民意、汇聚民智、凝聚民气，为解决广大群众提出的问题提供了重要的指导及意义。本文将基于自然语言处理和数据挖掘技术对群众问政留言记录及留言的答复意见进行深入的信息挖掘，从而提取我们所需要的更深度的信息。

针对问题一：通过正则表达式进行数据清洗，去除除中文以外得特殊字符。利用 jieba 中文分词工具对留言详情进行分词，然后利用 CountVectorizer 将文本中的词语转换为词频矩阵，使其向量化，构造三个差异性较大得分类器，朴素贝叶斯，KNN，SVM。获得三个初步训练好的分类器，然后逐条用三个分类器做预测，评估分类器。

针对问题二：本文根据附件 3 通过文本相似度计算法将特定时间、特定地点以及特定人群问题的留言进行归类，进而通过对文本量化、构建模型定义合理的热度评价指标，并给出评价结果，按表 1 的格式给出排名前五的热点问题，按表 2 的格式给出相应热点问题对应的留言信息。

针对问题三：

最后，我们根据目前“智慧政务”的实施方案与实际社情民意的反映需求的出入，提出针对性的建议。

关键词：去重 中文分词 K-means 聚类 贝叶斯算法 TF-IDF 算法

The Role of Text Mining in "Smart Government Affairs"

Abstract

With its advantages of high office efficiency, low communication cost, and simple management, the smart government system has now become an important channel for the government to conduct intelligent supervision, intelligent decision-making, and provide intelligent services to the public. Therefore, digging and analyzing the texts in "Smart Government Affairs" to understand public opinion, gather people's wisdom, and gather people's popularity, provides important guidance and significance for solving the problems raised by the masses. This article will conduct in-depth information mining based on natural language processing and data mining technology on the public's question and answer message records and message comments, so as to extract the more in-depth information we need.

Aiming at the problem of the first: The data is cleaned by regular expression to remove special characters except Chinese. Using the Chinese word segmentation tool of Jieba to segment the message details, then using countvectorizer to transform the words in the text into the word frequency matrix, making it vectorized, constructing three classifiers with great differences, naive Bayes, KNN, SVM. Three initially trained classifiers are obtained, and then three classifiers are used to predict and evaluate the classifiers one by one.

Aiming at the problem of the second: This article classifies the

messages of a specific time, specific place and specific group of people according to the text similarity calculation method according to Annex 3, and then defines a reasonable heat evaluation index by quantifying the text and building a model, and gives the evaluation results The top five hotspot questions are given in the format of Table 1, and the message information corresponding to the hotspot questions is given in the format of Table 2.

Aiming at the problem of the third, based on the current "smart government" implementation plan and the actual social conditions and public opinion reflecting the discrepancies in demand, we put forward targeted suggestions.

Keywords: deduplication Chinese word segmentation K-means clustering KNN algorithm Bayesian algorithm TF-IDF algorithm

目　录

1. 挖掘目标

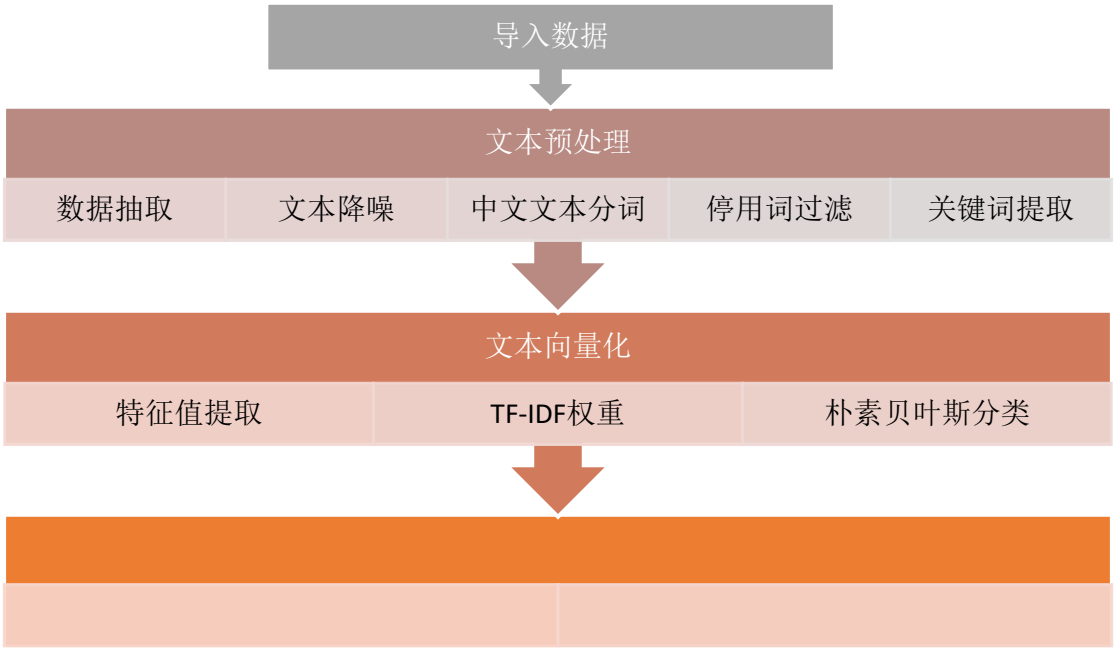
在现在信息爆炸的时代，网络信息量日益增加，许多微信、微博、市长信箱、阳光热线等网络平台成为了政府汇聚民智、凝聚民气的重要渠道，为了能更快更精准地对群众们反映的问题进行留言划分和分发部门，仅仅依靠人工是不行的。并在自然语言技术协同处理政务新趋势下，利用数据挖掘来减轻处理政务压力已成为一项亟待解决的问题。

本次的挖掘目标是利用政务平台上收集到的自互联网公开来源的群众问客留言记录，以及相关部门对部分群众留言的答复意见，数据源主要为文本数据，针对这种非结构化语言，我们首先进行的是数据的简单清洗、数据预处理、中文分词。

2. 总体流程与步骤

2.1 总体流程

本文的总体架构及思路如下：



2.2 总体步骤

步骤一：数据预处理，对附件 1、2、3、4 非结构化文本数据进行简单的数据清洗，如去除重复项、中文文本分词、停用词过滤、粗略的关键词提取等，便于后续数据挖掘与分析。

步骤二：文本向量化，为了已处理的文本数据导入后续的分类器中，基于 TF-IDF 权重算法提取关键词，构建词汇-文本矩阵，并标准化。随后构建分类器模型以及模型的评估，将多分类问题转化为二分类问题，利用 Python 提供的

sklearn 包，分别比较不同分类器的效果，并采用 F-Score 评估模型的好坏。
步骤三：文本聚类，相似度计算
步骤四：构建分类器，构造朴素贝叶斯、KNN、svc 分类器，查看不同分类器的效果
步骤五：分类器评价，运用 F-Score 评估模型的好坏，查看每个分类器的混淆矩阵。

问题一
问题一任务：根据附件 2 给出的数据，建立关于留言内容的一级标签分类模型。
问题一处理的数据：附件 2.xlsx
问题一挖掘难点：①文本语义带来的词语交叉

②多分类问题带来的难度

③数据不平衡带来的影响

④长文本的无意义表达

3. 文本分类

3.1 数据预处理

3.1.1 数据描述

本次的数据集共有四个文件，均为 excel 文件。数据量很大，约 9xxx 条文件，且某些待处理的数据文字较为冗杂、繁琐，必须经过一系列处理后才能使用。.

3.1.2 文本预处理

由于文本数据量很大并且部分文本数据过于冗杂，噪声过多，如果把数据不加以清洗，这会影响到后续分类器处理的精度，也会增大机器处理的负担。

因此，我们首先要进行文本预处理，主要包括数据抽取、文本降噪、中文分词、停用词过滤、词性标注、关键词提取五个部分。

（一） 数据抽取

通过对附件 2.xlsx 的不同留言类别的统计，我们得到留言类型中属于城乡建设/劳动与社会保障/教育问题/商贸旅游/环境保护/卫生

计生/交通运输分别为2009条/1969条/1589条/1215条/938条/877条/613条。显然这七个类别下的数据数量差距较大，比如城乡建设和劳动与社会保障的数据量已远超于属于交通运输的留言数，为了克服数据不平衡导致后续数据分类器的训练精度下降的不良影响。我们试验了多种数据抽取的方法以达到数据增强的效果，并最终采取了欠采样（又称下采样/under-sampling）的数据抽取方法。欠采样通过减少分类中多数类样本的样本数量来实现样本均衡，但会丢失多数类样本中的一些重要信息。

（二） 文本降噪

1. 去除特殊字符

正则表达式（Regular Expression），又称规则表达式，正则表达式是对字符串（包括普通字符（例如，a 到 z 之间的字母）和特殊字符（称为“元字符”））操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。正则表达式是一种文本模式，该模式描述在搜索文本时要匹配的一个或多个字符串。[1]

2. 去除内部重复词

很多时候，我们需要对语段中的单一词汇进行挖掘分析，相同的词汇并不会有太大价值。比如“哈哈哈哈哈”’今天天气天气天气天气天气真好好好好好好’等，过多的重复项对机器学习也会造成干扰，需及时去除。

（三） 中文分词

考虑到数据中存在大量冗杂、无意义无特征的语段，且中文语句属于非结构化语言，因此在对附件二.xlsx中的留言详情进行数据挖掘之前，我们先要把附件二.xlsx中的非结构化语言转化为计算机可以识别的结构化语言。我们首先将附件二表中的留言详情进行句子切割，即中文分词。但是中文句子中没有词的界限，因此在进行中文自然语言处理时会相对比较麻烦。这里我们采用了目前反映比较好的python中的中文分词包jieba进行分词。

jieba采用的算法有：

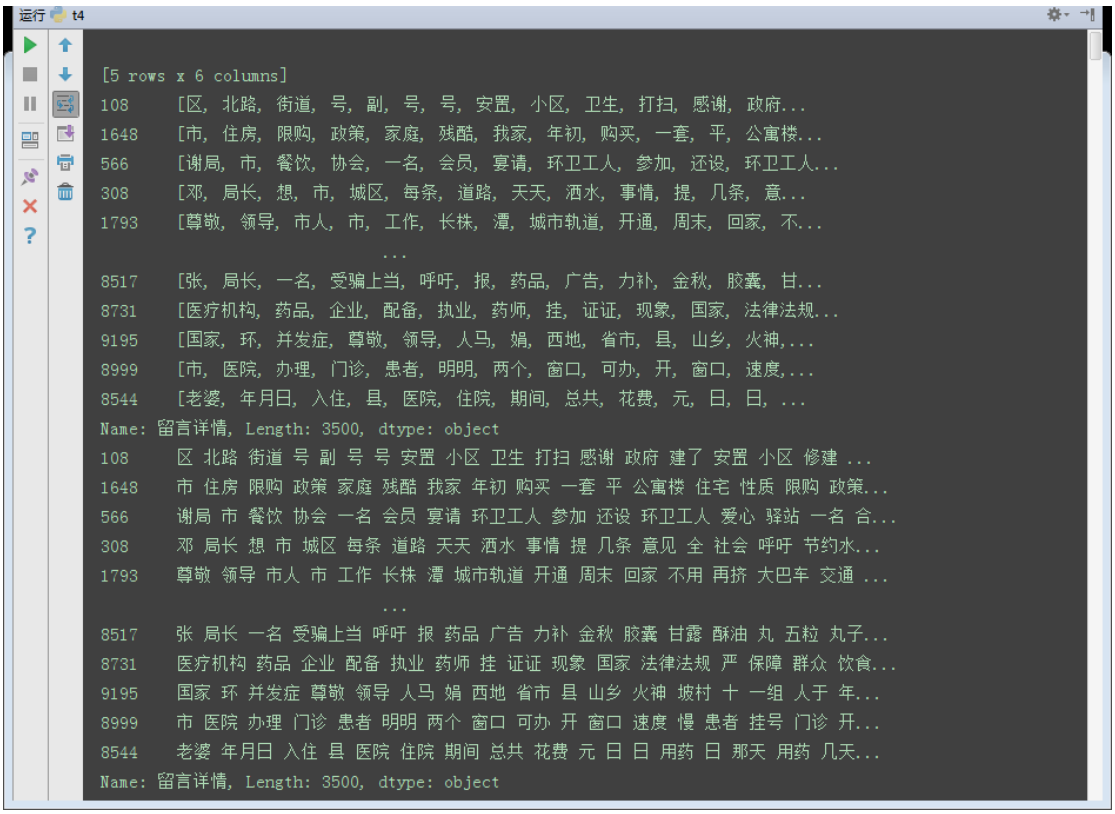
- 基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图（DAG）
- 采用动态规划查找最大概率路径，找出基于词频的最大切分组合
- 对于未登录词，采用了基于汉字成词能力的HMM模型，使用了Viterbi

算法

它支持三种分词模式

- 精确模式：试图将句子最精确地切开，适合文本分析；
- 全模式：把句子中所有的可以成词的词语都扫描出来，速度非常快，但是不能解决歧义；
- 搜索引擎模式：在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

以下是 jieba 处理后中文分词后的部分结果，见图 1：



中文分词后，我们发现语句普遍比较冗杂、很难提取到每个留言详情的特征词语，这是因为没有进行标点符号去除、停用词去除等数据清洗，可见噪声对文本分析造成的影响还很大。因此接下来会对数据进行进一步数据清洗。

（四）去除停用词

在信息检索与机器学习中，通常要在处理自然语言数据（或文本）之前或之后会自动过滤掉某些字或词，用以节省存储空间和提高搜索效率，这些字或词即被称为 Stop Words（停用词）。

通常，停用词大致分为两类。

一类是人类语言中包含的功能词，这些功能词极其普遍，与其他词相比，功能词没有什么实际含义，另一类词包括词汇词，比如'我们'等，这些词应用十分广泛，但是对这样的词搜索引擎无法保证能够给出真正相关的搜索结果，难以帮助缩小搜索范围，同时还会降低搜索的效率，所以通常会把这些词从问题中移去，从而提高搜索性能。

停用词：诸如“的”“是”“了”等常用词无任何意义，挖掘不到有效信息，反而会增加文本数据的噪声，也需要剔除。

对经过分词操作后的留言详情内容再进行删除停用词操作：通过观察原始数据，发现有很多如“的”“是”“了”这样的无用内容，这些都被称之为停用词，我们利用停用词表对分词后的文本数据进行处理，下 展示了经过分词和删除停用词两步预处理操作之后的部分内容：

```
Name: 留言详情, Length: 3500, dtype: object
108     区 北路 街道 号 副 号 号 安置 小区 卫生 打扫 感谢 政府 建了 安置 小区 修建 ...
1648    市 住房 限购 政策 家庭 残酷 我家 年初 购买 一套 平 公寓楼 住宅 性质 限购 政策...
566     谢局 市 餐饮 协会 一名 会员 宴请 环卫工人 参加 还设 环卫工人 爱心 驿站 一名 合...
308     邓 局长 想 市 城区 每条 道路 天天 洒水 事情 提 几条 意见 全 社会 呼吁 节约水...
1793    尊敬 领导 市人 市 工作 长株 潭 城市轨道 开通 周末 回家 不用 再挤 大巴车 交通 ...
                                ...
8517    张 局长 一名 受骗上当 呼吁 报 药品 广告 力补 金秋 胶囊 甘露 酥油 丸 五粒 丸子...
8731    医疗机构 药品 企业 配备 执业 药师 挂 证证 现象 国家 法律法规 严 保障 群众 饮食...
9195    国家 环 并发症 尊敬 领导 人马 娟 西地 省市 县 山乡 火神 坡村 十 一组 人于 年...
8999    市 医院 办理 门诊 患者 明明 两个 窗口 可办 开 窗口 速度 慢 患者 挂号 门诊 开...
8544    老婆 年月日 入住 县 医院 住院 期间 总共 花费 元 日 日 用药 日 那天 用药 几天...
```

（五）关键词提取[1]

关于文本的关键词提取方法分为有监督、半监督和无监督三种：

1、有监督的关键词抽取算法

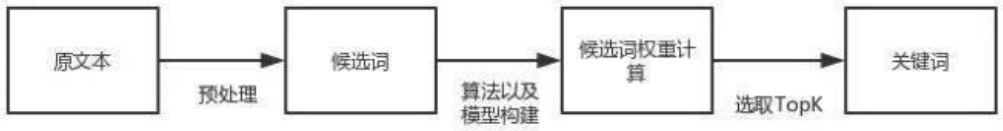
它是建关键词抽取算法看作是二分类问题，判断文档中的词或者短语是或者不是关键词。既然是分类问题，就需要提供已经标注好的训练语料，利用训练语料训练关键词提取模型，根据模型对需要抽取关键词的文档进行关键词抽取

2、 半监督的关键词提取算法

只需要少量的训练数据，利用这些训练数据构建关键词抽取模型，然后使用模型对新的文本进行关键词提取，对于这些关键词进行人工过滤，将过滤得到的关键词加入训练集，重新训练模型。

3、 无监督的方法

不需要人工标注的语料，利用某些方法发现文本中比较重要的词作为关键词，进行关键词抽取。



(六)文本特征抽取

经过噪声处理、停用词过滤等操作后，文本数据已较为规范。但由于包含了大量数据，加之文本数据为非结构化数据，机器难以处理。因此我们接下来就要考虑如何能够简洁的将一段语句中的核心关键词取出，关键词一般是指文本中出现频率较高且非无用的词语，其一定程度上代表了文本的语义核心所在。它既可以基本保留整个语句的大致语义，而且关键词少又方便我们后续的处理。查阅文献发现，目前比较认可的几种特征抽取的方法有词频—逆向文档频率(TF-IDF)、信息增益、互信息等。

词频-逆向文档频率(TF-IDF)[2]

TF-IDF(Term Frequency-Inverse Document Frequency)是一种用于资讯检索与资讯探勘的常用加权技术。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。

Document Frequency)。TF 表示一个单词在文档集合 d 中出现的频率。IDF 的基本想法为：如果包含项 t 的文档很少，那么我们可以给予一个高的重要性给它，这样一来他的 idf 就会很大，也就是说这个单词 s 对文本分类来说很重要。那么我们可以假设现在有一堆文档由集合 w 表示，其中包含单词 s 的文档的数量是 i，并且包含 s 的其他类的文档的总数是 j，显然所有包含 s

的文档数为 n(n=i+j)，当 i 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 s 类别区分能力不强。
下面是传统的 TF-IDF 公式：

$$W_j = TF_j * IDF$$

其中 W_j是文档中 Term 的权重，术语频率（TF）是指给定单词出现在文件中的频率。 对于特定文档中的单词，其重要性可表示为：

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

其中 ，其中$n_{i,\ j}$ 是 Term 在该篇文章中的出现次数，$\sum_k n_{k,\ j}$,是这篇文章的总词数。TF 是 Term 出现在所有文档中的频率，IDF 是 DF 的倒数,一般来说 IDF 的计算公式

$$IDF_i = \lg \frac{|D|}{|\{j:t_i \in d_j\}|}$$

对于本次数据而言，如果某个词很重要，它应该在这篇文章中多次出现。于是，我们进行"词频"（Term Frequency，缩写为 TF）统计。即在词频的基础上，要对每个词分配一个"重要性"权重。这个权重叫做"逆文档频率"（Inverse Document Frequency，缩写为 IDF），它的大小与一个词的常见程度成反比。即比较常见的词（”中国”）的权重就比较小；比较少见的词（"肺炎 ”）的权重就比较大。权重大的词在文中出现，我们就认为该词对该文本的影响较大，预测主题的能力就越强，可以体现文本的核心语义。

3.2 文本特征抽取

3.2.1 文本的向量化表示

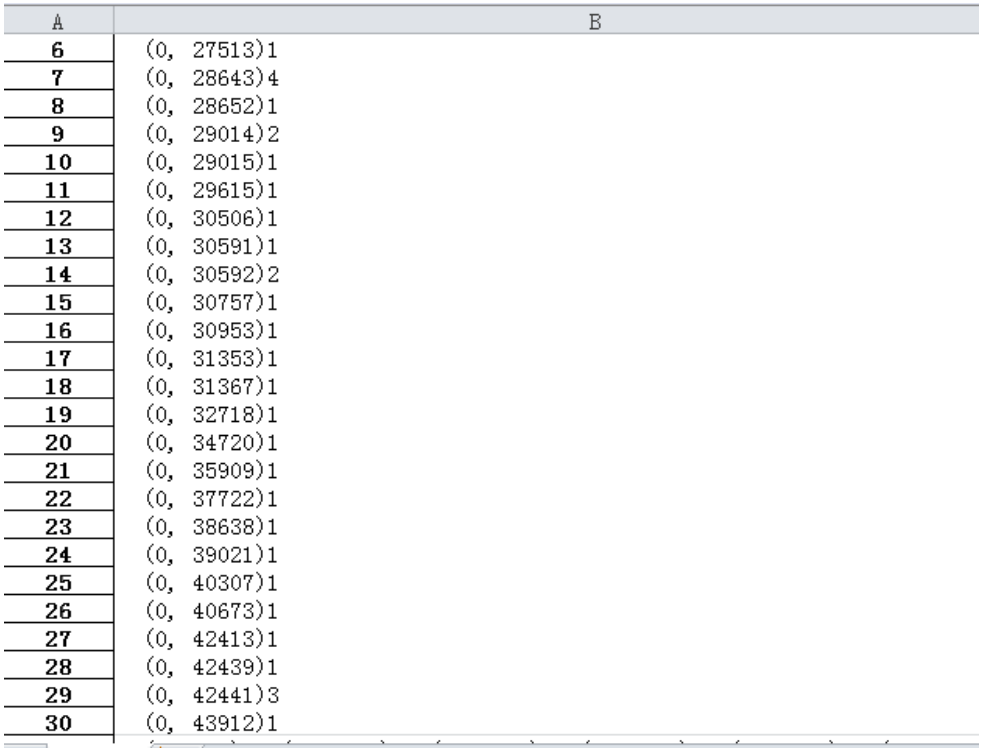
由于中文是非结构化语言，直接导入机器学习中的分类器中会难以处理。因此在分类之前，必不可少的要对文本进行向量化处理，在这里我们运用到 python 中的 CountVectorizer，它会将文本中的词语转换为词频矩阵。

3.2.2 词频向量化[3]

词袋模型(Bag of Words，简称 BoW)，即将所有词语装进一个袋子里，不考虑其词法和语序的问题，即每个词语都是独立的，把每一个单词都进行统计，同时计算每个单词出现的次数。也就是说，词袋模型不考虑文本中词与词之间的上下文关系，仅仅只考虑所有词的权重，而权重与词在文本中出现的频率有关。

按照传统步骤，运用词袋模型一般要经历分词（tokenizing）、统计修订词特征值（counting）与标准化（normalizing）三个步骤。具体操作如下：
词袋模型首先会进行分词，在分词之后，通过统计每个词在文本中出现的次数，我们就可以得到该文本基于词的特征，如果将各个文本样本的这些词与对应的词频放在一起，就是我们常说的向量化。向量化完毕后一般也会使用 TF-IDF 进行特征的权重修正，再将特征进行标准化。 再进行一些其他的特征工程后，就可以将数据带入机器学习算法进行分类聚类了。

在本次挖掘中， 我们直接在 python 环境下用 scikit-learn 的 CountVectorizer 类来完成词频向量化的任务，这个类可以帮我们完成文本的词频统计与向量化。以下是部分词频向量化的结果

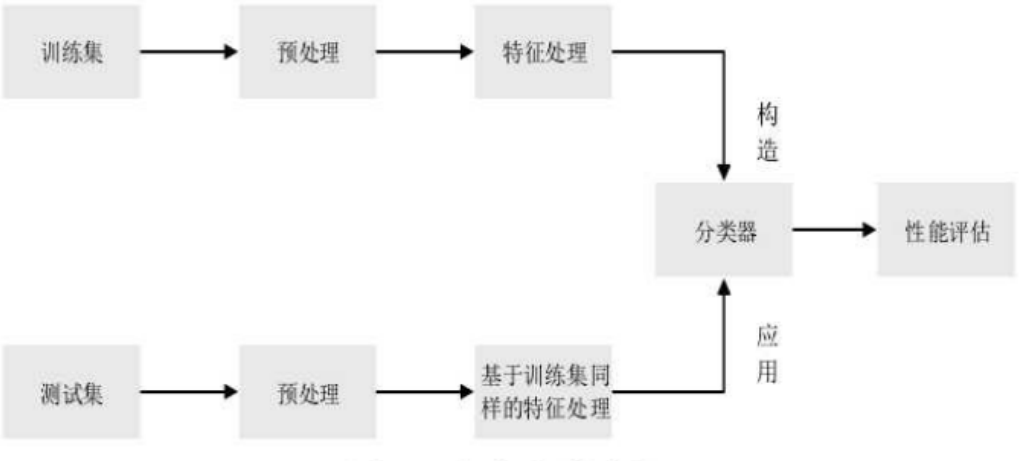


A	B
6	(0, 27513)1
7	(0, 28643)4
8	(0, 28652)1
9	(0, 29014)2
10	(0, 29015)1
11	(0, 29615)1
12	(0, 30506)1
13	(0, 30591)1
14	(0, 30592)2
15	(0, 30757)1
16	(0, 30953)1
17	(0, 31353)1
18	(0, 31367)1
19	(0, 32718)1
20	(0, 34720)1
21	(0, 35909)1
22	(0, 37722)1
23	(0, 38638)1
24	(0, 39021)1
25	(0, 40307)1
26	(0, 40673)1
27	(0, 42413)1
28	(0, 42439)1
29	(0, 42441)3
30	(0, 43912)1

A	B	C	D	E	F
6	(0, 1048)1 (0, 1595)1 (0, 2065)1 (0, 2415)1 (0, 3465)1 (0, 4244)1 (0, 5666)1 (0, 6329)1 (0, 6331)1 (0, 10				
7	(0, 42)1 (0, 9250)1 (0, 10124)1 (0, 15767)2 (0, 16827)1 (0, 20684)1 (0, 20880)1 (0, 21301)1 (0, 21951)1 (				
8	(0, 423)1 (0, 555)1 (0, 3120)1 (0, 4401)1 (0, 5305)1 (0, 5404)1 (0, 5405)1 (0, 5441)1 (0, 5442)2 (0, 6266				
9	(0, 182)1 (0, 568)1 (0, 1361)1 (0, 1532)1 (0, 1773)1 (0, 2050)4 (0, 2065)1 (0, 2927)1 (0, 3014)1 (0, 3085				
10	(0, 407)1 (0, 1354)1 (0, 1575)1 (0, 2065)2 (0, 2278)1 (0, 2883)1 (0, 2992)1 (0, 3061)1 (0, 3086)1 (0, 319				
11	(0, 93)1 (0, 424)1 (0, 1706)1 (0, 2209)1 (0, 2322)1 (0, 2362)1 (0, 2364)1 (0, 2376)1 (0, 2391)1 (0, 4102)				
12	(0, 1268)4 (0, 3401)2 (0, 3605)1 (0, 3705)1 (0, 4287)1 (0, 5683)1 (0, 5703)1 (0, 6130)1 (0, 7106)1 (0, 91				
13	(0, 3600)1 (0, 5830)1 (0, 9266)1 (0, 10124)2 (0, 14182)1 (0, 15356)2 (0, 16552)1 (0, 17137)1 (0, 17549)1				
14	(0, 1329)2 (0, 3242)1 (0, 6329)2 (0, 7025)1 (0, 8340)1 (0, 8506)1 (0, 9730)1 (0, 22312)1 (0, 22507)1 (0,				
15	(0, 343)1 (0, 559)1 (0, 632)1 (0, 1519)1 (0, 2065)3 (0, 2073)1 (0, 2269)1 (0, 3438)1 (0, 4102)2 (0, 4645)				
16	(0, 67)1 (0, 76)1 (0, 77)1 (0, 669)1 (0, 2654)1 (0, 4103)1 (0, 4432)1 (0, 4831)3 (0, 7071)3 (0, 7470)1 (				
17	(0, 376)1 (0, 1202)1 (0, 1575)1 (0, 2159)1 (0, 2168)1 (0, 3401)1 (0, 3745)1 (0, 4728)1 (0, 5749)1 (0, 575				
18	(0, 241)1 (0, 1766)1 (0, 2043)1 (0, 4244)1 (0, 7788)1 (0, 11283)1 (0, 14182)1 (0, 15604)1 (0, 16616)1 (0,				
19	(0, 471)2 (0, 520)1 (0, 628)1 (0, 1620)1 (0, 3926)1 (0, 4442)1 (0, 5330)1 (0, 7024)1 (0, 8340)1 (0, 11393				
20	(0, 4549)1 (0, 9730)1 (0, 15116)1 (0, 19260)1 (0, 19275)1 (0, 19471)1 (0, 24342)1 (0, 27033)1 (0, 27274)1				
21	(0, 251)1 (0, 632)1 (0, 640)1 (0, 868)2 (0, 1471)2 (0, 1532)1 (0, 1848)1 (0, 2065)17 (0, 2925)1 (0, 2979)				
22	(0, 254)1 (0, 882)1 (0, 3477)1 (0, 5099)1 (0, 6864)1 (0, 7036)1 (0, 9603)1 (0, 10317)1 (0, 10845)1 (0, 13				
23	(0, 310)2 (0, 1631)1 (0, 1665)1 (0, 2040)2 (0, 2065)9 (0, 2066)1 (0, 2762)1 (0, 3014)1 (0, 3199)1 (0, 330				
24	(0, 79)1 (0, 424)1 (0, 979)1 (0, 988)1 (0, 1250)1 (0, 1304)1 (0, 1452)1 (0, 1705)1 (0, 1784)1 (0, 2065)3				
25	(0, 2823)1 (0, 3953)1 (0, 5441)1 (0, 7339)1 (0, 11683)1 (0, 13417)1 (0, 13913)1 (0, 15356)2 (0, 15738)1 (				
26	(0, 1329)1 (0, 1408)1 (0, 5756)1 (0, 5836)1 (0, 7770)1 (0, 9300)1 (0, 19313)1 (0, 20601)1 (0, 23091)2 (0,				
27	(0, 296)1 (0, 386)3 (0, 1538)7 (0, 2050)2 (0, 2065)6 (0, 2654)1 (0, 3748)2 (0, 3749)1 (0, 4102)2 (0, 4126				
28	(0, 110)1 (0, 561)2 (0, 1045)1 (0, 1280)1 (0, 8017)2 (0, 8059)1 (0, 14309)1 (0, 14778)1 (0, 14809)1 (0, 1				
29	(0, 423)1 (0, 2040)1 (0, 2065)3 (0, 5700)2 (0, 5825)1 (0, 7036)1 (0, 7126)1 (0, 7339)1 (0, 8805)2 (0, 103				
30	(0, 1170)1 (0, 1450)1 (0, 1550)1 (0, 1612)1 (0, 3557)1 (0, 4441)1 (0, 4562)1 (0, 5683)1 (0, 5774)1 (0, 708				

3.2.3 文本分类

经过一系列预处理后，我们开始构建分类模型，首先我们将数据按照 60%、40%的比例分为训练集和测试集，训练集用来模型的训练和优化，测试集则用来测试分类器的实际应用中的表现。图是我们关于这次文本分类的流程图：



本次数据挖掘要求我们进行留言的分类，通过后续了解，我们发现这是个多分类问题。对于多分类问题，我们考虑先转化为多个二分类问题，再进行逐步分类，进而建立分类的一级模型。

目前处理分类问题的方法很多，比如神经网络、决策树、支持向量机、朴素贝叶斯等。对于文本分类而言，朴素贝叶斯分类（NBC）是以贝叶斯定理为基础并且假设特征条件之间相互独立的方法，它的的学习效率和分类效果都比较理想。

本次分类，我们基于 python 尝试了三种不同的分类器：朴素贝叶斯分类、支持向量机(SVM)、(KNN)，建立了相关的一级标签分类模型，并比较了其准确度，进而选出最优方案。

1、朴素贝叶斯[4]

朴素贝叶斯分类(NBC)算法的基本思想是：先通过已给定的训练集，以特征词之间独立作为前提假设，学习从输入到输出的联合概率分布，再基于学习到的模型，输入 X 求出使得后验概率最大的输出 Y。

设有样本数据集 D=$\{d_1, d_2, \ldots, d_n\}$，对应样本数据的特征属性集为 X=$\{x_1, x_{2\ldots\ldots} x_d\}$类变量 Y=$\{y_1, y_2, \ldots, y_m\}$,即 D 可以分为$y_m$类别。其中相互独立且随机，则 Y 的先验概率$P_{prior}$=P（Y），Y 的后验概率$P_{post}$=P$(Y|X)$，由朴素贝叶斯算法可得，后验概率可以由先验概率$P_{prior}$=P（Y）,证据 P（X），类条

$$P(Y|X) = \frac{P(Y)\,P(X|Y)}{P(X)}$$

件概率 P($X|Y$)计算出：

朴素贝叶斯基于各特征之间相互独立，在给定类别为 y 的情况下，上式可以进一步表示为下式：

$$P(X|Y=y) = \prod_{i=1}^{d} P(x_i|Y=y)$$

由以上两式可以计算出后验概率为：

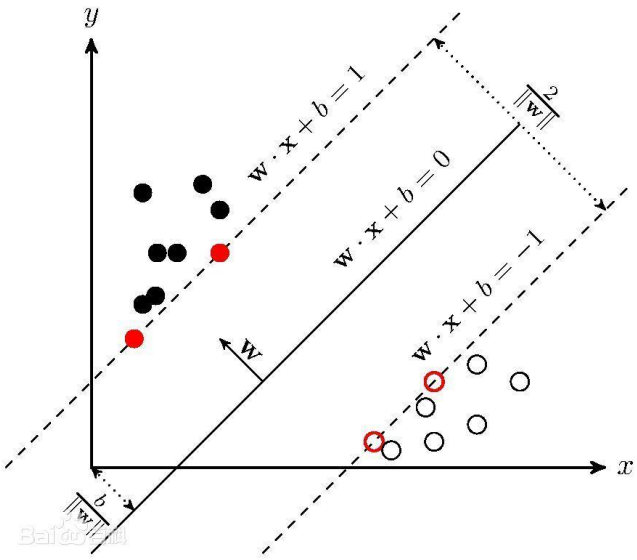
$$P_{post} = P(Y|X) = \frac{P(Y)\prod_{i=1}^{d} P(x_i|Y)}{P(X)}$$

由于 P(X)的大小是固定不变的，因此在比较后验概率时，只比较上式的分子部分即可。因此可以得到一个样本数据属于类别y_i的朴素贝叶斯计算如下图所示

$$P(y_i|x_1, x_2, \cdots, x_d) = \frac{P(y_i)\prod_{j=1}^{d} P(x_j|y_i)}{\prod_{j=1}^{d} P(x_j)}$$

2、多分类 SVM（**multiple class SVM**）[5]

SVM 是一种按监督学习方式的二分类模型。作为一种数据挖掘领域常用的一种机器学习算法,它凭借着有效防止过拟合的特性在分类领域得到了广泛应用。它包含三种模型：线性可分支持向量机/线性支持向量机/非线性支持向量机。SVM 学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示，$\omega x + b = 0$即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。



标准 SVM 是基于二元分类问题设计的算法，无法直接处理多分类问题。利用标准 SVM 的计算流程有序地构建多个决策边界以实现样本的多分类，通常的实现为“一对多（one-against-all）”和“一对一（one-against-one）”。一对多 SVM 对 m 个分类建立 m 个决策边界，每个决策边界判定一个分类对其余所有分类的归属；一对一 SVM 是一种投票法（voting），其计算流程是对 m 个分类中的任意 2 个建立决策边界，即$\frac{m(m-1)}{2}$共有个决策边界，样本的类别按其对所有决策边界的判别结果中得分最高的类别选取 。一对多 SVM 通过对标准 SVM 的优化问题进行修改可以实现一次迭代计算所有决策边界。

在获得文本数据后，首先对所有样本的特征数据进行归一化，再随机选取总样本数据的 60%作为训练数据集进行训练，剩下的 40%数据作为测试集用来评价 SVM 分类器的分类效果。

3、KNN 算法[6]

K 最近邻(kNN，k-NearestNeighbor)分类算法是数据挖掘分类技术中最简单的方法之一。在 KNN 算法中，每个样本都可以用它最接近的 k 个邻近值来代表。

考虑到本次处理的数据是可以分为若干类别，即一个样本在特征空间中的 k 个最相邻的样本中的大多数属于某一个类别，并具有这个类别上样本的特性。故在 Python 上尝试了用 KNN 分类器来完成此次分类。

以下是 KNN 的有关算法：

假设有训练数据集{X(i),Y(i)}i=1,…,N，其中每个样本{X(i),Y(i)}有 D 维属性 X(i)=(x1(i),x2(i),…,xD(i))以及一个属性值（或类别标号）Y(i)=y(i).

对于待预测的样本点 X(i)，定义其与训练集样本点{X(i),Y(i)}的属性距离为欧式距离（也可以是任何意义上的距离）：

$$d_{ji}=d(X^{(j)},X^{(i)})=\sqrt{\sum_{l=1}^{D}(x_l^{(j)}-x_l^{(i)})^2}$$

令标号 i 取遍 i=1,…,N，得到集合

$$d_j=\{d_{j,1},d_{j,2},\cdots,d_{j,N}\}$$

dj 是待预测样本点 X(j)与其他所有训练集样本点的距离集合.假设 dj 中数值最小的 K 个距离分别为{dj,s1,…,dj,sk}，KNN 算法选择其对应的 K 个训练集样本点$\{X^{(s_k)},Y^{(s_k)}\}k=_{1,\cdots,k}$对待预测样本点 X(j)对应的属性值 Y(j)进行预测，如果 y(j)是数值，一般取均值作为属性值 Y(j)的估计值

$$\hat{Y}^{(j)}=(1/K)\cdot\sum_{k=1}^{K}y^{(s_k)}$$

经过三次不同分类器的处理，可以看出对于本次数据挖掘，朴素贝叶斯分类器的效果最好，如图

accuracy			0.86	700
macro avg	0.85	0.85	0.85	700
weighted avg	0.86	0.86	0.86	700

accuracy			0.48	700
macro avg	0.64	0.49	0.50	700
weighted avg	0.66	0.48	0.50	700

accuracy			0.85	700
macro avg	0.85	0.85	0.85	700
weighted avg	0.86	0.85	0.85	700

3.2.4 模型评价

此次挖掘，我们最终将附件二中的测试文本集数据导入分类器进行分类后，

采用 F-Score 作为评判分类器模型的好坏程度。常用的分类器性能评估指标有：精确率（查准率）、召回率（查全率）和 F1。

精确率（查准率）：对于给定的已知类别的测试数据集，分类器对其进行正确分类的样本数与总样本数之比。

$$p_i=\frac{TP}{TP+FP}$$

召回率（查全率）：对于给定的已知类别的测试数据集，指分类器正确判断该类的样本个数与属于该类的实际样本总数之比。

$$R_i=\frac{TP}{TP+FN}$$

F1 值：假定查全率和查准率同等重要，即 F1 为查准率和查全率的调和平均。

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

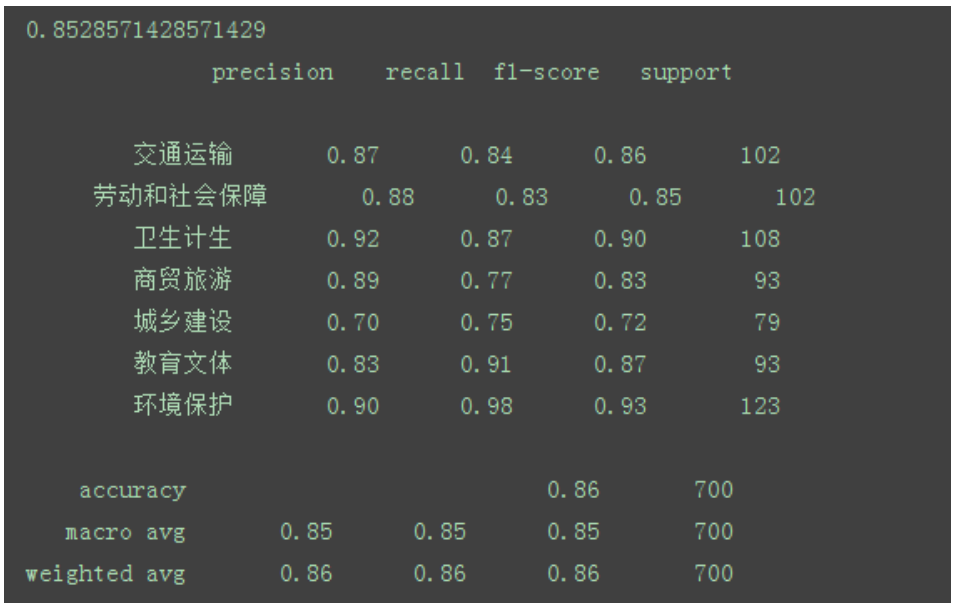
其中 P_i为第 i 类的查准率，R_i为第 i 类的查全率

另外我们也可以用另一种形式去描述 F-Score 评估方法

真实值	预测值	
	正例	反例
正例	TP(真正例)	FN(假反例)
反例	FP(假正例)	TN(真反例)

图 1 混淆矩阵

最终我们得到三种分类器对数据的效果对比差异，如图



	precision	recall	f1-score	support
交通运输	0.68	0.52	0.59	102
劳动和社会保障	0.61	0.50	0.55	102
卫生计生	0.66	0.60	0.63	108
商贸旅游	0.24	0.82	0.37	93
城乡建设	0.49	0.27	0.34	79
教育文体	0.91	0.45	0.60	93
环境保护	0.91	0.24	0.38	123
accuracy			0.48	700
macro avg	0.64	0.49	0.50	700
weighted avg	0.66	0.48	0.50	700

	precision	recall	f1-score	support
交通运输	0.89	0.82	0.86	102
劳动和社会保障	0.80	0.83	0.82	102
卫生计生	0.87	0.88	0.88	108
商贸旅游	0.89	0.81	0.85	93
城乡建设	0.66	0.80	0.72	79
教育文体	0.92	0.91	0.92	93
环境保护	0.92	0.89	0.91	123
accuracy			0.85	700
macro avg	0.85	0.85	0.85	700
weighted avg	0.86	0.85	0.85	700

4. 热点问题分析

4.1. 数据预处理

通过观察所给数据，要想及时发现热点问题，首先我们需要根据附件 3 将某一时段内反映特定地点特定人群问题的留言进行分类，而附件 3 中的“留言主题”和“留言详情”字段均含有多余的文本格式，需要将其量化成数值形式才能对其进行分析，并且留言主题与详情信息中存在噪声特征，如果把这些数据也一起引入进行分词、词频统计甚至文本聚类等等，则必然会对聚类结果的质量造成

很大的影响，从而本文首先需要对数据进行预处理。

4.1.1 文本预处理

我们把这些文本数据的预处理分为两个部分：

（一）数据清洗

首先从 excel 输入文件中读取数据，数据大概 20 多万行，然后通过 pandas 获取 excel 里的信息，最后通过分析数据问题，采用不同的方法将不同的脏数据变为可用数据。

①excel 表中的重复值：pandas 中有两个函数是专门用来处理重复值的，第一个是 duplicated 函数。Duplicated 函数用来查找并显示数据表中的重复值。

说明：第一，数据表中两个条目间所有列的内容都相等时 duplicated 才会判断为重复值。(Duplicated 也可以单独对某一列进行重复值判断)。第二，duplicated 支持从前向后(first)和从后向前(last)两种重复值查找模式，默认是从前向后进行重复值的查找和判断。

第二个是 Pandas 中的 drop_duplicates 函数用来删除数据表中的重复值，判断标准和逻辑与 duplicated 函数一样。使用 drop_duplicates 函数后，python 将返回一个只包含唯一值的数据表。

② excel 表中的空值/缺失值：在 python 中空值被显示为 NaN，Pandas 中查找数据表中空值的函数有两个，一个是函数 isnull，如果是空值就显示 True。另一个函数 notnull 正好相反，如果是空值就显示 False。

对于空值有两种处理的方法：第一种是使用 fillna 函数对空值进行填充，可以选择填充 0 值或者其他任意值。第二种方法是使用 dropna 函数直接将包含空值的数据删除。

③ excel 表中数据间的空格：第一种是去除数据两边的空格，第二种是单独去除左边的空格，第三种是单独去除右边的空格。

④ 数据中的异常和极端值：对数据进行描述性统计，使用 describe 函数可以生成描述统计结果。

（二）中文分词及去停用词

词：最小的能够独立活动的有意义的语言成分

-英文单词之间以空格分界

-汉语以字为基本书写单位，词语之间没有明确区分

分词：将连续的字序列按照一定的规范重新组合成词序列

本文我们采用的中文分词工具包是 Python 分词库：jieba

jieba 分词系统提供分词、词性标注、未登录词识别，支持用户自定义词典，关键词提取等功能。它有三种模式：

精确模式：试图将句子最精确的切开，适用于文本分析任务；

全模式：找出所有可以成词的词语，速度快，但不能解决歧义；

搜索模式：精确模式基础上，对长词再切分，适合用于搜索引擎分词；

在本文中，jieba 分词要首先自定义词典以及排除信息，这样效果会差异很大，然后形成一个二维数组，使用 gensim 中的 corpora 模块，将分词形成后的二维数组生成词典。

4.2 文本特征提取与选择

经过上述文本预处理后，虽然已经去掉部分停用词，但还是包含大量词语，给文本

向量化过程带来困难，而特征提取是指描述对象的属性不一定反映潜在的规律或模式，对属性进行重新组合，获得一组反映事物本质的少量的新的属性的过程。从而从属性集合中选择那些重要的、与分析任务相关的子集的过程。有效的特征选择不仅降低数据量，提高分类模型的构建效率，有时还可以提高分类准确率。所以特征提取的主要目的是在不改变文本原有核心信息的情况下尽量减少要处理的词数，以此来降低向量空间维数，从而简化计算，提高文本处理的速度和效率。特征选择步骤：

①根据一定的方法选择一个属性子集；

② 衡量子集的相关性；

③ 判断是否需要更新属性子集，若是，转第 1 步继续；若否，进入下一步；

④ 输出最终选取的属性子集 。

第一步方法：逐步增加法（stepwise forward selection）、逐步递减法（stepwise backward elimination）、随机选取。

第二步方法：一类称为 filter 方法，利用距离、信息熵以及相关度检验等方法直接衡量属性子集与类别的关联；另一类称为 wrapper 方法，利用分类模型来衡量属性子集的效果，通常效率很低。

文本特征选择的常用方法有词频-逆向文档频率(TF-IDF)、互信息、信息增益、X2 统计等。

在本文中，我们接着上述文本预处理后，使用字典将二维数组通过 doc2bow 稀疏向量生成语料库，刚开始使用 TF 模型算法，后来更改为：LsiModel 模型算法，将语料库计算出 TF-IDF 值。

（一）词频-逆向文档频率(TF-IDF)

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。

TF-IDF 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。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。

在一份给定的文件里，词频（term frequency，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(term count)的归一化，以防止它偏向长的文件。（同一个词语在长文件里可能会比短文件有更高的词数，而不管该词语重要与否。）对于在某一特定文件里的词语来说，它的重要性可表示为：

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

以上式子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（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，再将得到的商取对数得到：

$$\mathrm{idf}_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

其中|D|：语料库中的文件总数：包含词语的文件数目（即的文件数目）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以下式子作为分母。

$$1 + |\{d \in D : t \in d\}|$$

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 倾向于过滤掉常见的词语，保留重要的词语。

（二）互信息(Mutual Information)

互信息是信息论里一种有用的信息度量，它可以看成是一个随机变量中包含的关于另一个随机变量的信息量，或者说是一个随机变量由于已知另一个随机变量而减少的不肯定性。

在统计语言模型中，互信息用于表示两变个量间(表征 f 和类别 c 之间)的相关性。

其互信息记作 MI(f, c)可由下式计算：

$$\mathrm{MI}(f,c) = \log\left(\frac{p(f,c)}{p(c)p(f)}\right)$$

互信息没有考虑单词发生的频度，这是互信息一个很大的缺点，它导致互信息评估函数经常倾向于选择稀有词。

（三）信息增益

在信息增益中，衡量标准是看特征能够为分类系统带来多少信息，带来的信息越多，该特征越重要。对一个特征而言，系统有它和没它时信息量将发生变化，而前后信息量的差值就是这个特征给系统带来的信息量。所谓信息量，就是熵。

假如有变量 X，其可能的取值有 n 种，每一种取到的概率为 Pi，那么 X 的熵就定义为

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log p(x_i).$$

也就是说 X 可能的变化越多，X 所携带的信息量越大，熵也就越大。

对于文本分类或聚类而言，就是说文档属于哪个类别的变化越多，类别的信息量就越大。所以特征 T 给聚类 C 或分类 C 带来的信息增益为 IG(T)=H(C)-H(C|T)。

H(C|T)包含两种情况：一种是特征 T 出现，标记为 t，一种是特征 T 不出现，标记为 t’。所以 H(C|T)=P(t)H(C|t)+P(t’)H(C|t ‘)，再由熵的计算公式便可推得特征与类别的信息增益公式。

信息增益最大的问题在于它只能考察特征对整个系统的贡献，而不能具体到某个类别上，这就使得它只适合用来做所谓“全局”的特征选择（指所有的类都使用相同的特征集合），而无法做“本地”的特征选择（每个类别有自己的特征

集合，因为有的词，对这个类别很有区分度，对另一个类别则无足轻重)。

（四）χ^2 统计

χ^2 统计方法度量词条 t 和文档类别之间的相关程度，并假设 t 和 c 之间符合具有一阶自由度的 χ^2 分布。令 N 表示训练语料中的文本总数，c 为某一特定类别，t 表示特定的词条，A 表示属于 c 类且包含 t 的文档频数，B 表示不属于 c 类但是包含 t 的文档频数，C 表示属于 c 类但是不包含 t 的文档频数，D 是既不属于 c 也不包含 t 的文档频数，则对于 c 的 χ^2 值由下式计算：

$$\chi^2(t,c)=\frac{N\times(AD-CB)^2}{(A+C)(B+D)(A+B)(C+D)}$$

词条对于某类的 χ^2 统计值越高，它与该类之间的相关性越大，携带的类别信息也较多。

4.3 文本的词频矩阵

由于计算机不能够直接处理文本信息，因而我们需要对文本进行处理，将文本表示成为

计算机能够直接处理的形式，即文本数字化。文本表示)也称为文本特征表达，它不仅要求能够真实准确的反映文档的内容，而且要对不同的文档具有区分能力。

本文采用一种简单实用的主题模型——潜在语义索引(Latent Semantic Indexing,简称 LSI)，有的文章也叫 Latent Semantic Analysis（LSA)。LSI 是基于奇异值分解（SVD）的方法来得到文本的主题的。

SVD：对于一个 m×n 的矩阵 A，可以分解为下面三个矩阵：

$A_{m\times n}=U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}A_{m\times n}=U_{m\times m}\Sigma_{m\times n}V_{n\times n}{}^T$

有时为了降低矩阵的维度到 k，SVD 的分解可以近似的写为：

$A_{m\times n}\approx U_{m\times k}\Sigma_{k\times k}V^T{}_{k\times n}A_{m\times n}\approx U_{m\times k}\Sigma_{k\times k}V_{k\times n}{}^T$

如果把上式用到我们的主题模型，则 SVD 可以这样解释：我们输入的有 m 个文本，每个文本有 n 个词。而 A_{ij} 则对应第 i 个文本的第 j 个词的特征值，这里最常用的是基于预处理后的标准化 TF-IDF 值。k 是我们假设的主题数，一般要比文本数少。SVD 分解后，U_{i1} 对应第 i 个文本和第 l 个主题的相关度。V_{jm} 对应第 j 个词和第 m 个词义的相关度。Σ_{lm} 对应第 l 个主题和第 m 个词义的相关度。也可以反过来解释：我们输入的有 m 个词，对应 n 个文本。而 A_{ij} 则对应第 i 个

词档的第 j 个文本的特征值,这里最常用的是基于预处理后的标准化 TF-IDF 值。k 是我们假设的主题数，一般要比文本数少。SVD 分解后，U_{i1} 对应第 i 个词和第 l 个词义的相关度。V_{jm} 对应第 j 个文本和第 m 个主题的相关度。Σ_{lm} 对应第 l 个词义和第 m 个主题的相关度。这样我们通过一次 SVD，就可以得到文档和主题的相关度，词和词义的相关度以及词义和主题的相关度。

另外，通过 doc2bow 函数计算测试数据的稀疏向量。gensim 库 corpora 包的 dictionary 模块提供了文本数据结构化处理的一系列工具，其中 Dictionary 是 dictionary 模块下定义的类，可以实现词汇和词汇 id 之间的映射，即词典，该类的方法 doc2bow 可以将文本词汇集合转换为词袋模型表示形式，以列表形式返回，列表元素为 (词汇 id , 词频)的元组。要将原始文档转换为词频矩阵，首先要将各文档分词，从字符串转化为单词列表，再统计各文档单词，生成词典(Dictionary)，最后利用词典方法 doc2bow 将文档转化成词频表示的向量。

Gensim 是一款开源的第三方 Python 工具包，用于从原始的非结构化的文本中，无监督地学习到文本隐层的主题向量表达。它支持包括 TF-IDF，LSA，LDA，和 word2vec 在内的多种主题模型算法，支持流式训练，并提供了诸如相似度计算，信息检索等一些常用任务的 API 接口。

语料（Corpus）：一组原始文本的集合，用于无监督地训练文本主题的隐层结构。语料中不需要人工标注的附加信息。在 Gensim 中，Corpus 通常是一个可迭代的对象（比如列表）。每一次迭代返回一个可用于表达文本对象的稀疏向量。

向量（Vector）：由一组文本特征构成的列表。是一段文本在 Gensim 中的内部表达。

稀疏向量（Sparse Vector）：通常，我们可以略去向量中多余的 0 元素。此时，向量中的每一个元素是一个(key, value)的 tuple。

模型（Model）：是一个抽象的术语。定义了两个向量空间的变换（即从文本的一种向量表达变换为另一种向量表达）。

对文本向量的变换是 Gensim 的核心。通过挖掘语料中隐藏的语义结构特征，我们最终可以变换出一个简洁高效的文本向量。在 Gensim 中，每一个向量变换的操作都对应着一个主题模型。

4.4 文本聚类

4.4.1 文本相似度计算

相似度是用来衡量文本间相似程度的一个标准。在文本聚类中，需要研究文本个体

将的差异大小，也就是需要对文本信息进行相似度计算，将根据相似特性的信息进行归类。 文本相似度计算方法分为有监督和无监督两类。

有监督方法，就是用朴素贝叶斯分类器之类的有监督模型来判断文本相似性或者计算相似度。这类方法要求有一定数量的标注语料，构建的代价比较高；由于训练语料通常无法做得很大，模型的泛化性不够，实际用起来会有点麻烦；距离计算环节的复杂度会比较高。

无监督方法，就是用欧氏距离等方法，直接计算文本之间的距离或者相似度。这类方法的特点是：不需要标注语料，特征工程或者参数估计可以使用很大的数据；很多方法对语言的依赖比较小，可以应对多语种混杂的场景；距离计算环节复杂度较低。

在上面我们通过 LSI 得到的文本主题矩阵可以用于文本相似度计算。而计算方法一般是通过余弦相似度。

余弦相似度，又称为余弦相似性，是通过计算两个向量的夹角余弦值来评估他们的相似度。余弦相似度将向量根据坐标值，绘制到向量空间中，如最常见的二维空间。

余弦相似性通过测量两个向量的夹角的余弦值来度量它们之间的相似性。0 度角的余弦值是 1，而其他任何角度的余弦值都不大于 1；并且其最小值是-1。从而两个向量之间的角度的余弦值确定两个向量是否大致指向相同的方向。两个向量有相同的指向时，余弦相似度的值为 1；两个向量夹角为 90° 时，余弦相似度的值为 0；两个向量指向完全相反的方向时，余弦相似度的值为-1。这结果是与向量的长度无关的，仅仅与向量的指向方向相关。余弦相似度通常用于正空间，因此给出的值为-1 到 1 之间。

注意这上下界对任何维度的向量空间中都适用，而且余弦相似性最常用于高维正空间。例如在信息检索中，每个词项被赋予不同的维度，而一个维度由一个向量表示，其各个维度上的值对应于该词项在文档中出现的频率。余弦相似度因此可以给出两篇文档在其主题方面的相似度。

两个向量间的余弦值可以通过使用欧几里得点积公式求出：

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta.$$

给定两个属性向量，A 和 B，其余弦相似性 θ 由点积和向量长度给出，如下所示：

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}.$$

这里的A_i, B_i分别代表向量 A 和 B 的各分量。

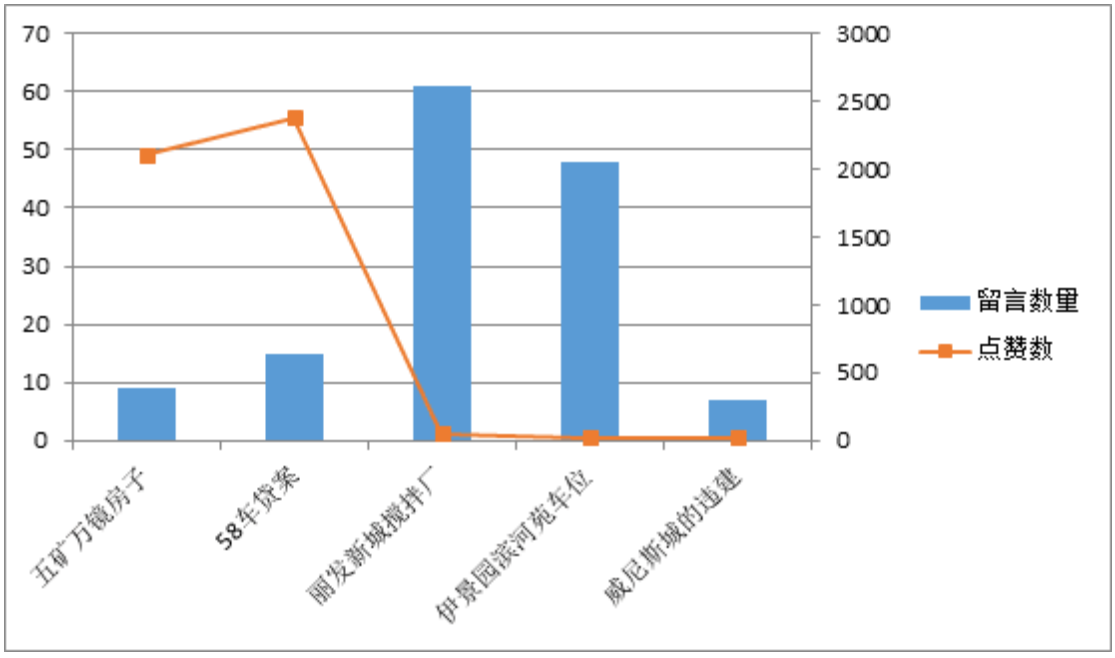
给出的相似性范围从-1 到 1：-1 意味着两个向量指向的方向正好截然相反，1 表示它们的指向是完全相同的，0 通常表示它们之间是独立的，而在这之间的值则表示中间的相似性或相异性。

对于文本匹配，属性向量 A 和 B 通常是文档中的词频向量。余弦相似性，可以被看作是在比较过程中把文件长度正规化的方法。

在信息检索的情况下，由于一个词的频率（TF-IDF 权）不能为负数，所以这两个文档的余弦相似性范围从 0 到 1。并且，两个词的频率向量之间的角度不能大于 90°。

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	2384	2019/1/11至2019/5/28	A市A4区	58车贷案
2	2	2109	2019/1/15至2019/11/11	A市五矿万境	A市A5区汇金路五矿万境K9县存在一系列问题
3	3	50	2019/4/16至2020/1/25	A2区丽发新城	搅拌站噪音扰民
4	4	18	2019/1/17至2019/12/7	A7县	威尼斯城大范围的违建
5	5	24	2019/7/7至2019/9/1	A市伊景园	房伊景园滨河苑销售若干问题的投诉

如图得到的前五留言数量和点赞数的对比：

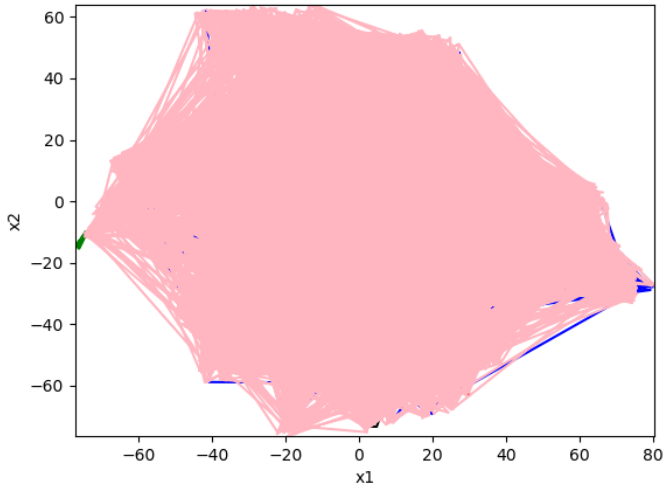


4.4.2 文本聚类

文本聚类（Text clustering），也称文档聚类，主要是依据著名的聚类假设：同类的文档相似度较大，而不同类的文档相似度较小。文本聚类就是将无类别标记的文本信息根据不同的特征，将有着各自特征的文本进行分类，使用相似度计算将具有相同属性或者相似属性的文本聚类在一起。作为一种无监督的机器学习方法，聚类由于不需要训练过程，以及不需要预先对文档手工标注类别，因此具有一定的灵活性和较高的自动化处理能力，已经成为对文本信息进行有效地组织、

摘要和导航的重要手段。因此可以通过文本聚类的方法对热点问题进行分类。通过聚类方法，可以对某一特定问题的答复具有更好的准确性。

聚类分布图，如图：



5. 答复意见分析

5.1 数据预处理

5.1.1 数据清洗

在题目给出的数据中，出现了很多无用的词语和字符，干扰了问题得分析，采取直接滤过方法，从文本中删除，因此首先导入 re 模块利用正则表达式清洗数据，然后引入自定义停用词，清洗后得数据保存至附件 4 去重.xlsx 中。

5.1.2 对留言答复意见的列表进行中文分词

在对留言的答复意见进行挖掘分析之前首先要把非结构化的文本信息转换为计算机能够识别的在对招聘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。这里采用 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。

5.2 相似性

相关性分析是指对两个或多个具备相关性的变量元素进行分析，从而衡量两个变量因素的相关密切程度，我们将群众投诉意见与政府对留言答复意见两者进行相似性处理。对于相似性处理，一般有四种方法，画图判断，皮尔逊相关系数，斯皮尔曼相关系数，余弦相关系数。

5.2.1 皮尔逊相关系数

皮尔逊相关系数要求样本满足正态分布 - 两个变量之间的皮尔逊相关系数定义为两个变量之间的协方差和标准差的商，其值介于-1 与 1 之间

皮尔逊相关系数具体原理如下：

协方差：$$s_{xy} = \frac{1}{n-1}\sum_{k=1}^{n}(x_k-\bar{x})(y_k-\bar{y})$$

标准差：$$s_x = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(x_k-\bar{x})^2}$$

皮尔逊相关系数：

$$\rho_{X,Y} = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

我们一般用欧式距离（向量间的距离）来衡量向量的相似度，但欧式距离无法考虑不同变量间取值的差异。举个例子，变量 a 取值范围是 0 至 1，而变量 b 的取值范围是 0 至 10000，计算欧式距离时变量 b 上微小的差异就会决定运算结果。而 Pearson 相关性系数可以看出是升级版的欧氏距离平方，因为它提供了对于变量取值范围不同的处理步骤。因此对不同变量间的取值范围没有要求（unit free），最后得到的相关性所衡量的是趋势，而不同变量量纲上差别在计算过程中去掉了，等价于 z-score 标准化。

3.1.3.2 斯皮尔曼相关系数

在统计学中，以查尔斯·斯皮尔曼命名的斯皮尔曼等级相关系数，即斯皮尔曼相关系数。

它是衡量两个变量的依赖性的非参数 指标。经常用希腊字母 ρ 表示。它利用单调方程评价两个统计变量的相关性。 如果数据中没有重复值， 并且当两个变量完全单调相关时，斯皮尔曼相关系数则为+1 或−1。

斯皮尔曼相关系数被定义成等级变量之间的皮尔逊相关系数。对于样本容量为 n 的样本，n 个原始数据被转换成等级数据，相关系数 ρ 为

$$\rho = \frac{\sum_i (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_i (x_i-\bar{x})^2 \sum_i (y_i-\bar{y})^2}}.$$

5.2.2 使用 numpy 计算协方差矩阵 相关系数

在进行文本挖掘的时候，我们不会把期望、方差、协方差一类的函数重新写一遍，在第三题中，使用了 numpy 类封装好的函数。这里使用 np.cov 函数，输出的结果是一个矩阵，这就是协方差矩阵。

接下来使用 numpy 计算相关系数

```
np.corrcoef(ab)
>> array([[ 1.        ,  0.18394201],
       [ 0.18394201,  1.        ]])
```

计算相关系数，我们使用 numpy 的 corrcoef 函数，这里的输出也是一个矩阵，这个矩阵数据的含义同上面的协方差类似，我们可以看到，这里我们的相关系数是 0.18 ，和我们上面自己编写的函数计算的结果一致。

5.3 相似匹配

项项是组成辞典的最基本单位,它用来描述事物或概念.项具有各种属性值(如对概念进行解释或说明）,项可以包含子项、子子项，它们之间可以是-种嵌套关系.

目的是将人民的投诉与政府的答复联系起来。对应的工作表元素首先在知识匹配字典 DOM 树上搜索（“消息响应”），然后根据特征值确定字典中央叶元素的位置关系。一般来说，可以有四种不同的情况。

（1）该材料的两个特性对应于同一片。例如，两相组装知识的特性值为‘45’。

（2）相同的父节点和相同的物理特性与不同的叶片元件相对应，如果叶片元件位于同一家长节点内，则材料特性的相似性是父节点元件的相似性值。两阶段组装知识的物质特征是“10”和“45”，如图 3 所示，它们具有相同的碳结构钢父母节点，其项目相似值为 0.9，即组装材料特性的相似性为 0.9。

（3）与同一个节点和祖先组分的特性相对应的工作表不在同一父系节点中，但可以在同一祖先节点中找到。计算相似性的方法包括：通过计算每个路径节点的“元素相似性”的结果，首先计算共同祖先节点的子节点与相应叶片元素（不包括共同祖先节点）之间的路径的语义相似性。通过将结果乘以共同祖先节点元件的相似性值和将知识特性组合成两个阶段的相似性来实现步骤 1 中计算的结果的最小值。如两相匹配装配知识的材料特征值分别为“70-G1”和“45”,根据图3,它们有共同的祖先结点“钢”,那么它们的相似度:①70-G1 ： Path1=弹簧钢\70-G1, SM(Path1)=1X0. 85=0.85; 45: Path2=碳素结构钢\45,SM(Path2)=1X0.9=0.9;②Min(0. 85,0.9)=0.85;③SM(70-G1,45)=0. 6X0.85=0. 51.

4)其他情况.在辞典中找不到对应项或没有共同的祖先结点,则认为其相似度为0..

5.4 Gensim

Gensim 是一款开源的第三方 Python 工具包，用于从原始的非结构化的文本中，无监督地学习到文本隐层的主题向量表达。它支持包括 TF-IDF，LSA，LDA，和 word2vec 在内的多种主题模型算法，支持流式训练，并提供了诸如相似度计算，信息检索等一些常用任务的 API 接口。

1 基本概念

语料（Corpus）：一组原始文本的集合，用于无监督地训练文本主题的隐层结构。语料中不需要人工标注的附加信息。在 Gensim 中，Corpus 通常是一个可迭代的对象（比如列表）。每一次迭代返回一个可用于表达文本对象的稀疏向量。

向量（Vector）：由一组文本特征构成的列表。是一段文本在 Gensim 中的内部表达。

稀疏向量（SparseVector）：通常，我们可以略去向量中多余的 0 元素。此时，向量中的每一个元素是一个(key, value)的元组

模型（Model）：是一个抽象的术语。定义了两个向量空间的变换（即从文本的一种向量表达变换为另一种向量表达）。

2 步骤一：训练语料的预处理

由于 Gensim 使用 python 语言开发的，为了减少安装中的繁琐，直接使用 anaconda 工具进行集中安装，

输入：pip install gensim，这里不再赘述。

训练语料的预处理指的是将文档中原始的字符文本转换成 Gensim 模型所能理解的稀疏向量的过程。

通常，我们要处理的原生语料是一堆文档的集合，每一篇文档又是一些原生字符的集合。在交给 Gensim 的模型训练之前，我们需要将这些原生字符解析成 Gensim 能处理的稀疏向量的格式。由于语言和应用的多样性，我们需要先对原始的文本进行分词、去除停用词等操作，得到每一篇文档的特征列表。

接下来，我们可以调用 Gensim 提供的 API 建立语料特征（此处即是 word）的索引字典，并将文本特征的原始表达转化成词袋模型对应的稀疏向量的表达。

到这里，训练语料的预处理工作就完成了。我们得到了语料中每一篇文档对应的稀疏向量（这里是 bow 向量）；向量的每一个元素代表了一个 word 在这篇文档中出现的次数。值得注意的是，虽然词袋模型是很多主题模型的基本假设，这里介绍的 doc2bow 函数并不是将文本转化成稀疏向量的唯一途径。在下一小节里我们将介绍更多的向量变换函数。

最后，出于内存优化的考虑，Gensim 支持文档的流式处理。我们需要做的，只是将上面的列表封装成一个 Python 迭代器；每一次迭代都返回一个稀疏向量即可。

3 步骤二：主题向量的变换

对文本向量的变换是Gensim的核心。通过挖掘语料中隐藏的语义结构特征，我们最终可以变换出一个简洁高效的文本向量。

在 Gensim 中，每一个向量变换的操作都对应着一个主题模型，例如上一小节提到的对应着词袋模型的 doc2bow 变换。每一个模型又都是一个标准的 Python 对象。下面以 TF-IDF 模型为例，介绍 Gensim 模型的一般使用方法。

首先是模型对象的初始化。通常，Gensim 模型都接受一段训练语料（注意在 Gensim 中，语料对应着一个稀疏向量的迭代器）作为初始化的参数。显然，越复杂的模型需要配置的参数越多。

```
from gensim import models
tfidf = models.TfidfModel(corpus)
```

其中，corpus 是一个返回 bow 向量的迭代器。这两行代码将完成对 corpus 中出现的每一个特征的 IDF 值的统计工作。

接下来，我们可以调用这个模型将任意一段语料（依然是 bow 向量的迭代器）转化成 TFIDF 向量（的迭代器）。需要注意的是，这里的 bow 向量必须与训练语料的 bow 向量共享同一个特征字典（即共享同一个向量空间）。

```
doc_bow = [(0, 1), (1, 1)]
print tfidf[doc_bow] # [(0, 0.70710678), (1, 0.70710678)]
```

注意，同样是出于内存的考虑，model[corpus]方法返回的是一个迭代器。如果要多次访问 model[corpus]的返回结果，可以先将结果向量序列化到磁盘上。

我们也可以将训练好的模型持久化到磁盘上，以便下一次使用：

```
tfidf.save("./model.tfidf")
tfidf = models.TfidfModel.load("./model.tfidf")
```

Gensim 内置了多种主题模型的向量变换，包括 LDA，LSI，RP，HDP 等。这些模型通常以 bow 向量或 tfidf 向量的语料为输入，生成相应的主题向量。所有的模型都支持流式计算。

4 步骤三：文档相似度的计算

在得到每一篇文档对应的主题向量后，我们就可以计算文档之间的相似度，进而完成如文本聚类、信息检索之类的任务。在 Gensim 中，也提供了这一类任务的 API 接口。

首先，我们需要将待检索的 query 和文本放在同一个向量空间里进行表达（以 LSI 向量空间为例）：

构造 LSI 模型并将待检索的 query 和文本转化为 LSI 主题向量

转换之前的 corpus 和 query 均是 BOW 向量

```
lsi_model = models.LsiModel(corpus, id2word=dictionary, num_topics=2)
documents = lsi_model[corpus]
```

query_vec = lsi_model[query]

接下来，我们用待检索的文档向量初始化一个相似度计算的对象：

index = similarities.MatrixSimilarity(documents)

我们也可以通过 save()和 load()方法持久化这个相似度矩阵：

index.save('/tmp/test.index')

index = similarities.MatrixSimilarity.load('/tmp/test.index')

注意，如果待检索的目标文档过多，使用 similarities.MatrixSimilarity 类往往会带来内存不够用的问题。此时，可以改用 similarities.Similarity 类。二者的接口基本保持一致。

最后，我们借助 index 对象计算任意一段 query 和所有文档的（余弦）相似度：

sims = index[query_vec]

返回一个元组类型的迭代器：(idx, sim)。

6. 结论

在本次实验的过程中，我们发现了许多问题与规律，总结了以下几点：

（1） 留言内容多变，但是主要内容如地区，时间，事件都会包含在其中，去除所有不重要的因素，得到的便是正确的结果。

（2） 语义带有歧义，比如“司法局的亲戚拖欠我们的钱”，表面上是与司法局有关，但是不然。这就需要我们更准确地分析问题。

智慧政务系统凭借其办公效率高、通信成本低、管理简便等优点，现已成为政府对民众进行智能监管、智能决策以及提供智能服务的重要渠道。因此，对“智慧政务”中的文本进行挖掘分析，了解民意、汇聚民智、凝聚民气，为解决广大群众提出的问题提供了重要的指导及意义。

由于能力的不足，我们未能完全的实现所有的问题，但是在这一次比赛中，我们获取了很多的

我们要通过看视频，看书来学习这些技术，更好的顺应时代的发展。大数据的快速发展，是的信息可以快速传播并广泛应用。一个人的思维和能力始终是有局限性的，众人拾柴火焰高，1+1+1>3,经过团队的分工协作，互相帮助，互相进步，才能完成这份实验。当今社会是团结的社会，需要集思广益的时代。

[1]张亚娜,高子婷,胡溢,杨成.融媒体新闻生产中的中文评论关键词提取[J].人工智能,2020(02):57-66.

[2] 刘擎权. 基于改进的 TFIDF 算法在文本分析中的应用[D].南昌大学,2019.

[3] 黄春梅,王松磊.基于词袋模型和 TF-IDF 的短文本分类研究[J].软件工程,2020,23(03):1-3.

[4] 张航. 基于朴素贝叶斯的中文文本分类及 Python 实现[D].山东师范大学,2018.

[5] Hsu, C.W. and Lin, C.J., 2002. A comparison of methods for multiclass support vector machines. IEEE transactions on Neural Networks, 13(2), pp.415-425
[6] 王轶凡.考虑数据时效性的高效 KNN 算法[J].赤峰学院学报(自然科学版),2019,35(11):19-21。
[7]朱嘉琪.讯问笔录相似问答对的匹配算法研究,2018.6
[8]]于成龙,于洪波.网络爬虫技术研究[J].东莞理工学院学报.201 1(03):25-29.
[9] BHARAT K, BRODER A. Mirror, miror on the Web: a study of host pairs with replicated
content[J] Computer Networks, 1999,31(11-16): 1579-1590.
[10]王鑫, 张亚男, 丘宏俊. 基于辞典的相似匹配[J]. 延边大学学报:自然科学版, 2007(03):43-46.