基于“智慧政务”文本数据挖掘

摘要

随着大数据时代的来临以及政府办公过程中积淀数据量的增长，政府部门成为了信息化时代主要的数据持有者之一。近年来，随着电子技术、信息技术和网络技术的发展，物联网、移动互联网和云计算等技术的日益发展，并推动了大数据的发展，这为以提供公共服务为核心的智慧政务带来了发展的技术基础和推动力量，利用数据分析相关技术来推动政府行政能力就显得尤为重要。

针对问题 1，对给定的数据进行中文分词，然后提取文本特征生成词向量，最后结合文本分类中常用的机器学习算法即支持向量机(SVM),K 近邻分类(KNN)和朴素贝叶斯(NB)进行分类，通过准确率和 F1 这两个指标对模型性能进行评价。根据 python 实验结果得出结论：SVM 模型的分类性能明显优于其他两种机器学习的分类算法，但所需训练时间也最多。KNN 的分类效果次之，NB 的分类效果相比之下最差。总体来说，三种分类算法的分类效果按照从优到劣可表示为：SVM>KNN>NB.

针对问题 2，即热点问题数据挖掘，首先,研究如何获得数据，对数据的真实性进行考量，本文采用的是题目给出的热点数据，所以真实性相对较好。其次，研究了附件 3 中文文本分词、文本向量化、文本聚类以及数据降维等数据处理技术，并通过这些技术将热点数据的文本型信息转换成数值型数据，之后通过 K 均值算法对其进行聚类分析;最后求得其分类结果，从结果显示 K-Means 聚类比层次聚类处理大量的文本数据较好。

针对问题 3，答复意见评价，本文采用如上问题 1 和问题 2 数据预处理方法，并在此基础上对答复意见进行相关性处理，得到政府相关部门对群众的热点留言答复意见的质量与情况。本文从结果上显示政府相关部门对热点问题答复的相关想不太理想，应加强答复群众留言的效率。

关键词：常用的机器学习算法；k-means 聚类算法；层次聚类

目录

1 挖掘目标

1.1 挖掘背景

智慧政务是指可以实现服务的高效化，数据实时化，响应的及时化的政务工具，可以简单、高效的解决百姓的各类问题，数据统计清晰明了化。随着以云计算、大数据、物联网为代表的信息技术快速发展,智慧政务的全面落实、完善，智慧政务的实现直接关系到我国的民生问题和社会稳定。近年来，国家一系列政策的实施，使得政务信息数据公开透明化，也为大数据研究政务信息提供了分析基础。

随着大数据时代的来临以及政府办公过程中积淀数据量的增长，政府部门成为了信息化时代主要的数据持有者之一。大数据时代，政府将自身所保有的部分数据向社会公众公开，促进了社会对这些公开数据进行增值利用以及创新应用，为社会的生产、生活以及社会经济活动服务，创造了不可估量的社会公共价值，如果能够有效利用好这些数据，海量的数据信息将会向社会释放出巨大的数据价值。对于人们来说，智慧政务信息进行深入的数据分析和挖掘可以为人们今后的参与问政提供重要导向和依据。对于政府来说，智慧政务会涉及到政府办公透明化，深入的分析、数据挖掘可以成为政府工作指南和政策改进的重要支撑，甚至可能成为政府职能部门决策的依据。因此，利用数据分析相关技术来推动政府行政能力就显得尤为重要。

首先，数据可以更加准确、全面、及时地反应出人们的客观现实和需求。智慧政务群众留言数据和留言回复体现了政府对人们反映民生问题解决方法和互动形式，进而为政府执政工作的改革和发展提供了更加可靠、及时、全面、客观的事实依据和素材。

其次，数据分析更加注重数据的开放性，一 方面，通过数据的公开，能够有效的促进政府工作过程的公开透明，进而强化社会的监督；另一方面，可以为人们反应日常生活问题提供有力依据以及有关部门深入挖掘数据的潜在价值，进而依据挖掘出的潜在规律调节自身行为，采取相应措施进行协调配合，可以为参与问政提供极大便利。

1.2 挖掘目标

根据附件 2 关于留言数据整理对其使用支持向量机（SVM），k 邻近分类（KNN）和朴素贝叶斯（NB）等方法进行分类处理，并使用“准确率”、“错误率”来衡量这几种分类算法优越程度。结合附件 3 的数据，对其进行特定地点和特点人群留言进行聚类，对其热点留言进行归类划分，构建热点留言问题明细，得到热点留言的综合评价指标和综合评价模型。

2 问题分析

2.1 问题 1 的分析

针对问题 1，对给定的数据进行中文分词，然后提取文本特征生成词向量，最后结合文本分类中常用的机器学习算法即支持向量机(SVM),K 近邻分类(KNN)和朴素贝叶斯(NB)进行分类，通过准确率，F1 两个指标对模型性能进行评价。

2.2 问题 2 的分析

针对问题 2，附件 3 的留言主题和留言详情运用相对应的文本相似度算法和聚类算法，确定热点问题评价指标。这里先对附件 3 进行文本分词、去除停用词、文本向量化处理等预处理操作，然后利用 k-means 聚类和主成分分析相结合的方法对每一个热点问题进行比较、分析以及评价。

2.3 问题 3 的分析

针对问题 3，首先对附件 4 进行数据处理，使其答复意见这一指标文本向量化，再使用相关性函数对其进行相关性分析。得到相关政府部门对群众留言的答复情况和答复质量相关性。从而推动智慧政务服务建设项目的发展，以及提高群众参与问政的积极性。

3 模型假设与符号说明

3.1 模型假设

1、假设题目所给的数据真实，可靠。
2、假设所给数据全都是中文文本，不存在英文字符。
3、假设热点留言问题分类和聚类 dp 由留言主题和留言详情指标体现。

3.2 符号说明

m	样本的数量	b	位移项
x_i	第 i 个样本	α_i	拉格朗日乘子
y_i	第 i 个样本的类别	ω	法向量
P	查准率	R	查全率
c_j	第 j 个簇的簇中心	n_j	第 j 个簇的样本总量

4 数据预处理

数据预处理主要分为三个部分，由于数据属于中文文本，它没有像英文的单词空格那样隔开，因此我们需要分词算法来完成分词。对中文文本分词会出现很多无效的词，比如“的”，“公司”，还有一些标点符号，这些对文本原意没有影响的词，我们应该去除。分词，去除停用词后我们都一般用 TF-IDF 函数对其进行文本向量化。这样我们才能为后续对附件 2 进行分类处理和聚类处理。

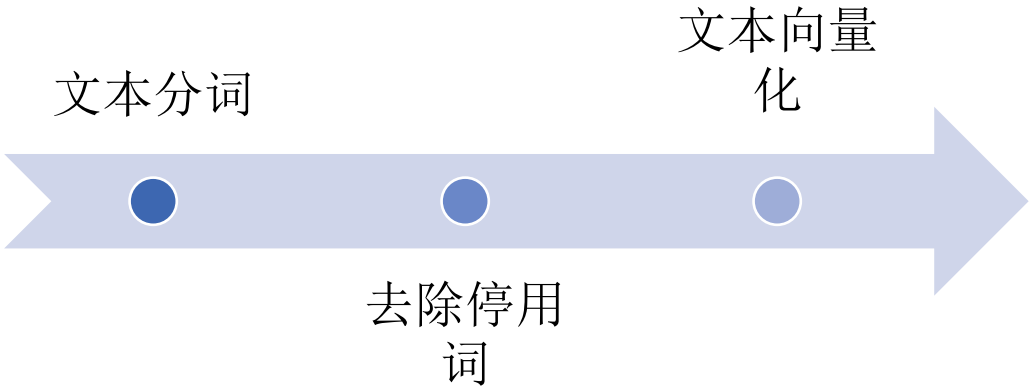

图 1 数据预处理流程图

4.1 文本分词

文本预处理是文本聚类的关键一步，文本预处理主要包括文本分词、去除停用词、文本向量化表示等步骤。

中文文本分词是将中文文档以中文中的最小单位词语来进行分割的操作，将句子分割成一个个的词语，但是分割后的词语集合能最大限度的表示原来长句子

的意思。文本分词基于英文的分词较为简单，英文文本中有空格作为词语的分割符号，以空格进行划分就可以完成，而中文文本中，没有类似于英文中的空格作为词语的分割界限，所以中文分词算法要比英文复杂很多。常见的分词算法大致有三类：基于理解的分词算法、基于统计的分词算法和基于字符串匹配的算法。

基于理解的分词算法，这种算法主要是让计算机来模拟人理解句子的过程，达到中文分词的目的。在这一过程中，会对中文文本进行语义和语法分析，利用语法和语义信息处理理解句子过程中的歧义，所以这种算法主要包括三个部分，分词系统，句法语义系统以及总控部分。由于中文语法语义的复杂性，很难将各式各样的语言信息组织成计算机可以直接读取的形式，所以基于理解的分词算法还仅仅是处于探索实验的阶段。

基于统计的分词算法，我们把经常出现的汉字的相邻组合称为词语。那么在文本中，相邻的两个汉字出现的次数越多就越有可能是一个词语，可以通过统计文本中相邻字共同出现的概率来判断这些字是否构成词语。这种分词算法的优点是分词速度快，但是缺点是会将一 些经常在一起出现的汉字、但却并不是词语的组合误认为词语进行切分。识别词语正确率不高。

基于字符串匹配的算法，这种算法是将字符串的子串和词典中的词进行匹配，如果找到相同的，就是词语，否则不是词语，这种匹配算法有正向和逆向两种匹配方法。这种算法由于已经存在词典，在切分过程中不需要考虑词义语义，所以切分速度较快，在分词算法中很常用。

本文使用的是结巴分词器进行分词，它是一个比较完善的中文分词器，使用的是基于正向最大字符串匹配的分词算法。

4.2 去除停用词

在中文分词结束后，分析数据特点发现，有一部分词对文本原意思的影响不大，比如“的”、“公司”、“省”、 “县”、“有限责任公司”等词对公司名称的影响不大，这些对文本原意没有贡献的词被称为停用词。停用词基本会出现在所有的文本中，这样无疑增加了文本之间的相似性，影响聚类结果，所以应该去除停用词。去除这些停用词可以有效减少处理数据的维度，提高处理数据的速度。对于这些需要去除的停用词，本文专门建立.了停用词表，在处理时，调用停用词表可以过滤掉文本中对应的词语。在分词过程中去除停用词的过程如图所示。

4.3 文本向量化

分词、去除停用词完成后，接着要做的是将文本进行向量化表示，以便进行后续的数学处理。

5 问题求解

5.1 群众留言分类

5.1.1 模型准备

对于分类模型中的评价指标。在分类模型中，我们通常会用“准确率”、“错误率”来衡量该算法模型的性能，这是分类任务中最常见的两种评价指标，准确率是分类正确的样本占总样本数量的比例，反之，则为错误率，对于数据集D，准确率可定义为

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\left(f(x_i)=y_i\right) \tag{1}$$

错误率可定义为

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\left(f(x_i)\neq y_i\right) \tag{2}$$

其中，m为数据集中样本的数量，x_i代表第i个样本，y_i为第i个样本的类别。

准确率和错误率虽然常用，但是也有它的局限性，并不能满足所有的分类任务中模型的度量和评价。因此，引入查准率、查全率和F_1值的概念，这三个概念在信息检索领域上有着广泛的使用。

将该问题转化为多个二分类问题，即类别为正或者反，预测结果和真实结果往往存在四种情况：实际上是正，预测为正：实际上为反，预测为正：实际上为正，预测为反;实际上为反，预测为反。以上四种情况分别记作：真正例(TP)，假正例(FP)，假反例(FN)，真反例(TN)，分类结果的混淆矩阵如表 1 所示。

表 1 分类结果的混淆矩阵

真实情况	预测情况	
	正例	反例
正例	TP	FN
反例	FP	TN

根据表 1，查准率P定义为

$$P=\frac{TP}{TP+FP} \tag{3}$$

查全率R定义为

$$R=\frac{TP}{TP+FN} \tag{4}$$

根据公式（3）和（4）可知，查准率 P 和查全率 R 是在很多情况下往往会矛盾，只有少数的情况下才能保证查准率 P 和查全率 R 同样很高，大部分情况往往是，查准率 P 很高，但查全率 R 比较低，而查全率 R 较高时，查准率 P 变低了，因此本文引入在机器学习分类任务中同样非常常用的模型度量评价指标 F_1 的概念。

F_1 是查准率 P 和查全率 R 的调和平均数，即

$$F_1=\frac{1}{2}\left(\frac{1}{P}+\frac{1}{R}\right)=\frac{2PR}{P+R} \tag{5}$$

由于涉及多个分类，故最终用来对分类方法进行度量评价的指标为

$$F_1=\frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i+R_i} \tag{6}$$

各类的词云图如下：

图 2 劳动和社会保障词云图

图 3 交通运输词云图

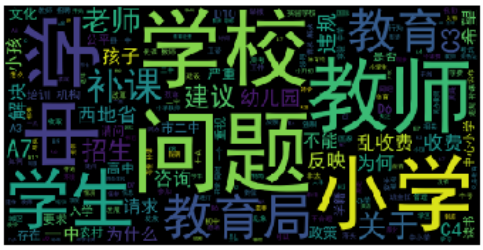

图 4 教育文体词云图

图 5 商贸旅游词云图

图 6 城乡建设词云图

图 7 环境保护词云图

图 8 卫生计生词云图

结合文本分类中常用的机器学习算法即支持向量机（SVM），k 近邻分类（KNN）和朴素贝叶斯（NB）进行分类，通过准确率，F1 等指标对模型性能进行评价。

5.1.2 支持向量机（SVM）

支持向量机(*SVM*)是机器学习中常见的监督学习方法，其在小样本、非线性分类问题中表现出了良好的性能而备受关注，SVM 使用该算法建立在统计学理论的基础上，在计算机视觉、自然语言处理及生物信息学等方面有广泛的应用。

SVM 的主要思想是基于训练集数据在空间中找到一个使得类别之间的间距最大的“超平面”，这样可以认为此时的分类效果最佳。用数学语言可表示为

训练集样本为 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，划分的超平面可以通过线性方程描述为 $\omega^T x+b=0$。其中，$\omega=(\omega_1,\omega_2,\cdots,\omega_d)$ 为法向量，决定该超平面的方向，位移项 b 决定超平面的具体位置，该平面记为 (ω,b)。

样本空间中任意的样本 x 到该平面的距离可以表示为

$$\gamma=\frac{\left|\omega^T x+b\right|}{\omega} \tag{7}$$

超平面如果能将所有的训练样本正确地分类，则有

$$\begin{aligned}\omega^T x_i+b\geq+1, y_i=+1\\ \omega^T x_i+b\geq-1, y_i=-1\end{aligned} \tag{8}$$

距离超平面最近的训练样本 x_i，即使得公式(8)成立的训练样本称之为“支持向量”，两个属于不同的类别的支持向量的距离定义为“间隔”，数学公式表示为

$$\gamma=\frac{2}{\|\omega\|} \tag{9}$$

支持向量机的基本原理就是使得“间隔”最大的超平面，也就是找到满足式子(8)的参数 ω 和 b，使得 γ 最大，即

$$\begin{aligned}&\max_{\omega,b}=\frac{2}{\|\omega\|}\\&s.t.y_i\left(\omega^T x+b\right)\geq 1,i=1,2,\cdots m\end{aligned}\tag{10}$$

公式(10)等价于

$$\begin{aligned}&\max_{\omega,b}=\frac{1}{2}\|\omega\|^2\\&s.t.y_i\left(\omega^T x+b\right)\geq 1,i=1,2,\cdots m\end{aligned}\tag{11}$$

显然，这是一个凸二次规划最优解求解问题，接下来对这一公式进行具体的推导演算。对式(11)使用拉格朗日乘子法可得到其“对偶问题”，对式(11)的每条约束添加拉格朗日乘子$\alpha_i\geq 0$，则该问题可写成

$$L\left(\omega,b,a\right)=\frac{1}{2}\|\omega\|^2+\sum_{i=1}^{m}\alpha_i\left(1-y_i\left(\omega^T x+b\right)\right)\tag{12}$$

其中$\alpha=\left(\alpha_1,\alpha_2,\cdots,\alpha_m\right)$，令$L\left(\omega,b,a\right)$对$\omega$和$b$的偏导为零，得到

$$\begin{aligned}&\omega=\sum_{i=1}^{m}\alpha_i y_i x_i\\&0=\sum_{i=1}^{m}\alpha_i y_i\end{aligned}\tag{13}$$

将(13)代入(12)，将$L\left(\omega,b,a\right)$中的ω和b消去，得到如下的对偶问题

$$\begin{aligned}&\max_{\alpha}\sum_{i=1}^{m}\alpha_i-\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j {x_i}^T x_j\\&s.t.\sum_{i=1}^{m}\alpha_i y_i=0\\&\alpha_i\geq 0,i=1,2,\cdots m\end{aligned}\tag{14}$$

解出 a 后，进而可求解得到模型

$$f\left(x\right)=\omega^T x+b=\sum_{i=1}^{m}\alpha_i y_i {x_i}^T x+b\tag{15}$$

在以上的研究和推导中，一直是假设训练集样本是线性可分的，但实际情况中，往往会出现很多线性不可分情况，因此，在支持向量机中引入核函数，来解决非线性的分类为题。

支持向量机中的核函数主要是处理异或问题、减少高维度中计算量巨大的问题。简而言之，核函数的作用就是将原始空间映射到一个合理的空间维度，同时使得训练集样本线性可分。

设$\phi\left(x\right)$为将x映射后的特征向量，在新的空间维度划分的超平面可以表示为

$$f\left(x\right)=\omega^{T}\phi\left(x\right)+b \tag{16}$$

进而有

$$\begin{aligned}&\max_{\omega,b}=\frac{2}{\left\|\omega\right\|}\\&s.t.y_{i}\left(\omega^{T}\phi\left(x\right)+b\right)\geq 1,i=1,2,\cdots m\end{aligned} \tag{17}$$

其对偶问题见(18)，此时如果对式子(18)进行求解，则涉及到需要计算$\phi\left(x_{i}\right)^{T}\phi\left(x_{j}\right)$，是映射之后的新的特征空间的内积，如果该特征空间的维度非常高，那么计算起来是非常复杂和困难的。

$$\begin{aligned}&\max_{\alpha}\sum_{i=1}^{m}\alpha_{i}-\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_{i}\alpha_{j}y_{i}y_{j}\phi\left(x_{i}\right)^{T}\phi\left(x_{j}\right)\\&s.t.\sum_{i=1}^{m}\alpha_{i}y_{i}=0\\&\alpha_{i}\geq 0,i=1,2,\cdots m\end{aligned} \tag{18}$$

如果有这样的一个函数

$$K\left(x_{i},x_{j}\right)=\left\langle\phi\left(x_{i}\right),\phi\left(x_{j}\right)\right\rangle=\phi\left(x_{i}\right)^{T}\phi\left(x_{j}\right) \tag{19}$$

则式子(18)的求解问题会简单很多，这里的()就是“核函数”，核函数的具体定理如下：

令χ为输入空间，$K\left(\because\right)$是定义在$\chi\times\chi$上的对称函数，当且仅当对于任意的数据$D=\left\{x_{1},x_{2},\cdots,x_{m}\right\}$，满足“核矩阵"始终是半正定的，$K$为核函数。

$$K=\begin{bmatrix}K\left(x_{1},x_{1}\right)&\cdots&K\left(x_{1},x_{j}\right)&\cdots&K\left(x_{1},x_{m}\right)\\\vdots&\ddots&\vdots&\ddots&\vdots\\K\left(x_{i},x_{1}\right)&\cdots&K\left(x_{i},x_{j}\right)&\cdots&K\left(x_{i},x_{m}\right)\\\vdots&\ddots&\vdots&\ddots&\vdots\\K\left(x_{m},x_{1}\right)&\cdots&K\left(x_{m},x_{j}\right)&\cdots&K\left(x_{m},x_{m}\right)\end{bmatrix}$$

通过计算得到用支持向量机分类准确率约为 89%，F_{1}值约为 0.89，运用支持向量机判定分类时，各类分类的准确率图 9 所示。

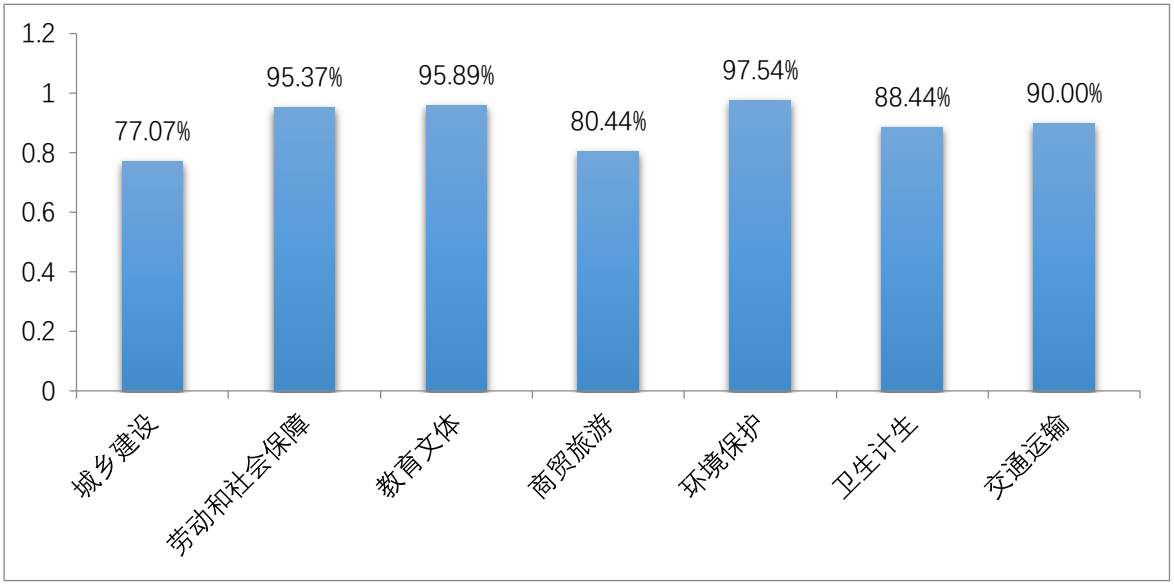

图 9 SVM 分类各类准确率

图 9 表明用支持向量机方法在各类中的准确率都较高，其中劳动和社会保障、教育问题、环境保护以及交通运输的准确率都高达 90%以上。

5.1.3 K 近邻分类（KNN）

K 近邻分类，也称 K 近邻分类算法，是机器学习中常用的最简单的分类算法之一。它的思路非常简单明了：如果在特征空间中一个样本周围的 k 个样本大多数都属于同一个类别，那么可以认为这个样本也属于这个类别。还可以基于距离的远近对其周围的 k 个样本的进行加权，距离越近，权重越大。

一般来说，k 通常是不大于 20 的整数。因为 KNN 是一种有监督学习算法，所以样本周围的 k 个“邻居”都是已经有标记类别信息的分类样本。其算法在训练过程中的思路可描述为：

（1）对数据进行前期预处理，提取文本特征并标注类别，存储训练数据和测试元组；

（2）计算测试数据与各个训练数据之间的距离，在文本分类问题中多采用余弦距离；

（3）设定参数 k，选取距离该测试数据距离最近的 k 个训练数据；

（4）计算这 k 个训练数据所标注类别出现的概率；

（5）返回这 k 个点出现频率最高的类别作为该测试数据的预测类别。

通过计算得到用 KNN 分类准确率约为 83%，F_1 值约为 0.83，运用 KNN 判定分类时，各类分类的准确率图 10 所示。

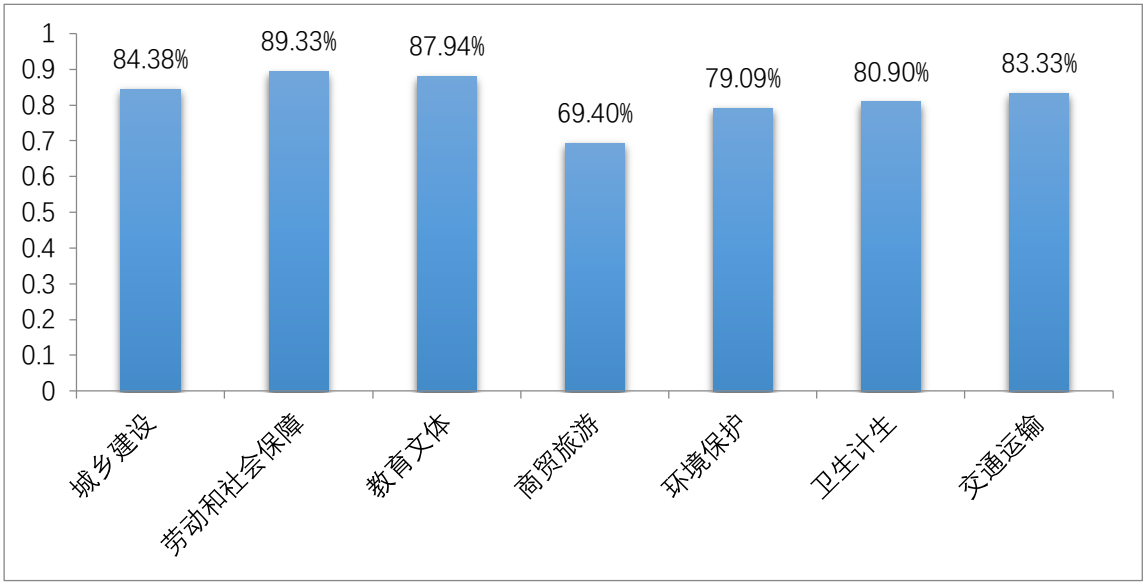

图 10 KNN 分类各类准确率

图 10 表明用 KNN 方法在各类中分类效果较好，其中劳动和社会保障、教育问题以及城乡建设的准确率分别为 89.33%、87.94%、84.38%。且最低的商贸旅游也为 69.40%。

5.1.4 朴素贝叶斯（NB）

在阐述朴素贝叶斯分类算法原理时，先对贝叶斯分类原理进行介绍。对于分类任务来讲，在所有相关概率都已知的情况下，贝叶斯分类是有限利用这些已知概率信息和最小的误判损失风险来进行分类的。

对于 N 类别分类问题，记类别为 $C=\{c_1,c_2,\cdots,c_N\}$，设 λ_{ij} 是将真实类别为 c_j 误分为 c_i 所产生的损失。基于后验概率 $P(c_i|x)$ 表示将样本 x 分类为 c_i 所产生的期望损失，即“条件风险”。

$$P(c_i|x)=\sum_{j=1}^{N}\lambda_{ij}P(c_j|x) \tag{20}$$

根据贝叶斯判定准则，最小化每个样本的条件风险，使得总体的条件风险最小，可以得到

$$h^*(x)=\arg\min_{c\in C}P(c|x) \tag{21}$$

此时，称 $h^*(x)$ 为贝叶斯最佳分类器。

如果根据最小化分类错误率的贝叶斯判别准则，选择使后验概率最大的类别进行标记，贝叶斯最佳分类器可表示为

$$h^*(x)=\arg\max_{c\in C} P(c|x) \tag{22}$$

其中 $P(c|x)$ 代表的是样本所有属性的联合概率。

如果从训练样本中直接计算是非常困难的，因此提出了朴素贝叶斯的算法。朴素贝叶斯算法是基于贝叶斯理论基础，假设训练样本的各特征条件相互独立的一种分类算法。基于各属性特征独立的假设，式子(22)可写为

$$P(c|x)=\frac{P(c)P(c|x)}{P(x)}=\frac{P(c)}{P(x)}\prod_{i=1}^{d}P(x_j|c) \tag{23}$$

其中，d 为属性数目，x_i 为样本 x 的第 i 个属性的取值。

因此，朴素贝叶斯的最佳分类器可写成

$$h_{nb}(x)=\arg\max_{c\in C} P(c)\prod_{i=1}^{d}P(x_j|c) \tag{24}$$

计算得分类准确率约为 75%，F_1 值约为 0.42，运用贝叶斯判定分类时，各类分类的准确率如图 11 所示。

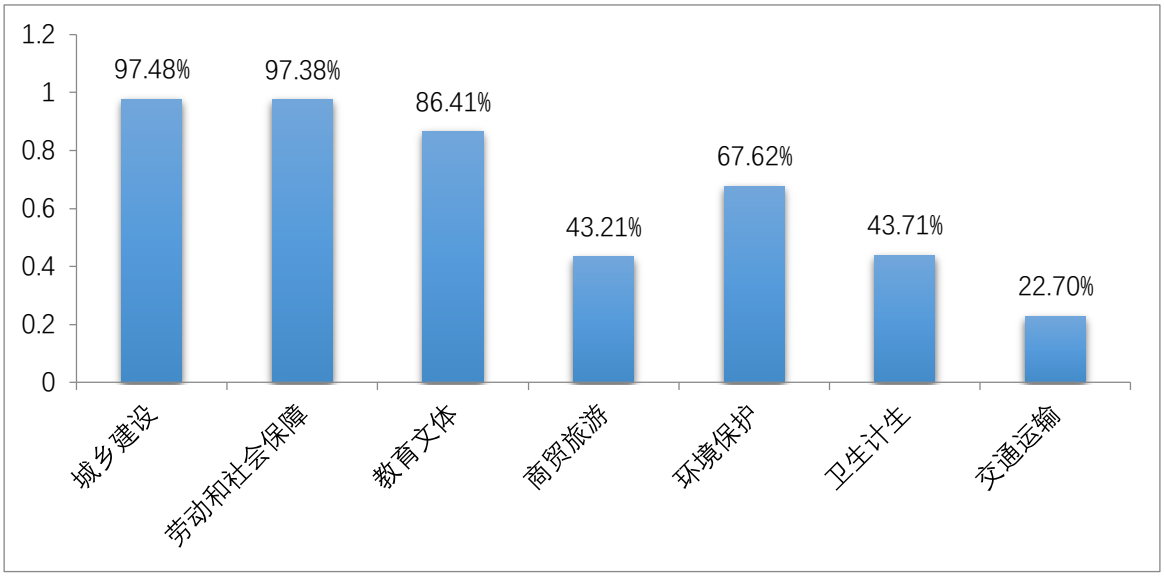

图 11 NB 分类各类准确率

图 11 表明朴素贝叶斯在城乡建设、劳动和社会保障以及教育文体中的分类效果较好，且准确率分别为 97.48%，97.38%以及 86.41。而在其他四类中的分类

效果并不是很好，且交通运输的准确率低至 22.7%。

5.1.5 模型比较

将三种模型中各分类的准确率进行比较，如图 12。

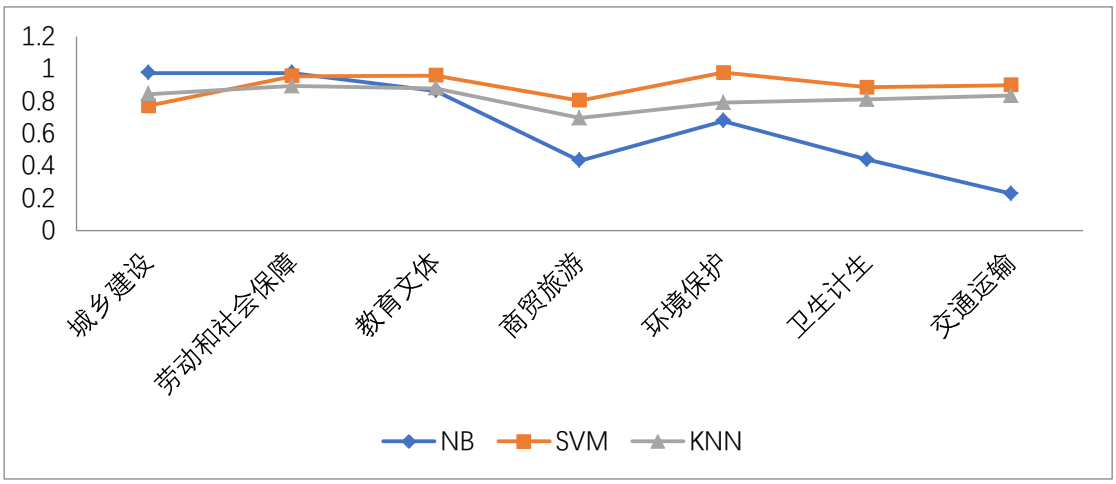

图 12 三种模型中各分类的准确率比较

将三种模型中总的准确率以及 F_1 值进行比较，如表 2。

表 2 三种模型中总的准确率以及 F1 值比较

分类模型	准确率	F_1 值
NB	0.746	0.416
SVM	0.887	0.885
KNN	0.831	0.826

由表 2 可知 SVM 方法只是在城乡分类中的效果不如其他两类，但是总体来说 SVM 的分类效果较其他两种方法更好。综上，分类性能排序为：SVM>KNN>NB

5.2 对热点问题挖掘与研究

5.2.1 模型的准备

本问主要研究内容是：针对附件 3 获取到的数据，建立聚类模型进行聚类分析。聚类分析主要的任务是将具有某种相同或者相似性质的物体划分为一类，以便进一步发现隐藏在数据中的信息。

在本文中，主要分析的是热点问题的挖掘，所谓的热点问题（某一时段内群众集中反映的某一问题可称为热点问题）本文采用基于 TF-IDF 和余弦相似度的短文本聚类算法对热点问题进行聚类，聚类结果将热点问题划分为若干类，有效的帮助相关部门对留言信息的整理，同时可以对热点问题进行热度排序，方便政府能够及时采取有效的举措对问题进行解决与回复，进而不断完善提高政府自身自身的业务能力。同时通过对聚类结果进行分析，也可以对有关部门进行政务监督，防止懒政现象的发生。分析结果同样可以帮助人们更好的参与问政的方式和方法。

文本热点问题挖掘过程图，如下图所示。

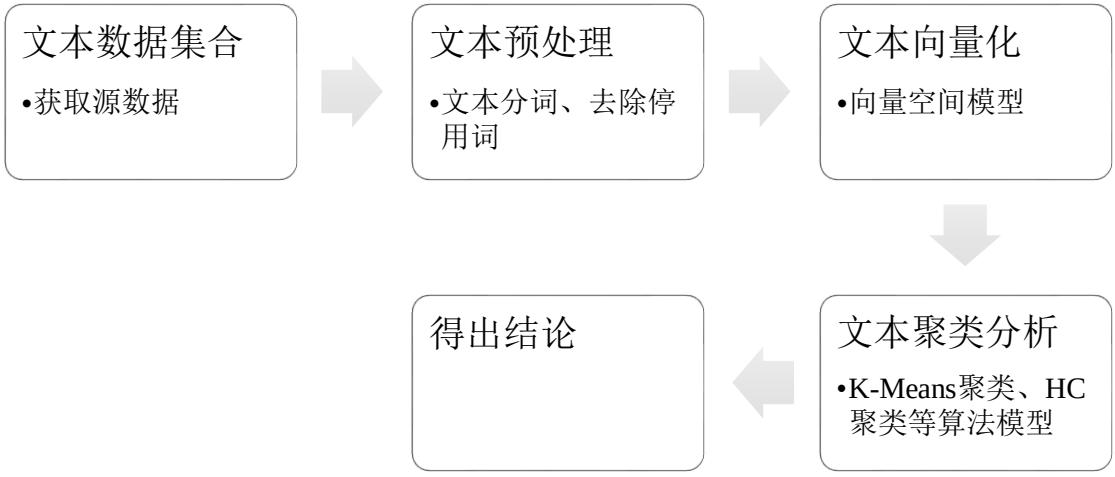

图 13 文本热点问题挖掘过程图

根据过程图中的处理步骤，本章中的各个小节将逐一介绍文本分析各个步骤中的技术以及在群众留言数据中的具体使用。

对附件 3 数据进行特点分析，不难发现，所有的数据都是中文文本数据，进行分析时，必须将文本数据转化成计算机可以识别处理的形式，形成可以处理的数学描述形式，通过正则表达式获取到的中标机构字段长度较短，而且在附件 3 文本中一些数据公共部分，这些部分在后续分析过程中没有作用，因此在预处理过程中会进行相关处理。同时在一些群众留言主题中，同一主题，反映的人不只一个，这样就使得一些数据长度较短，一些数据长度较长，数据之间长短差距较大，导致向量维数很大，这在聚类过程很容易引起聚类程序运行时间过长，运行效率低下的问题，在后期可视化过程中，维数过高也给可视化造成了一定的难度，所以在处理过程中对数据降维也是非常重要的一步。本文在数据预处理部分已经对文本数据进行如何分词，去除停用词已有详细说明，下面是对文本向量化进行具体分析，目前最常用的方法是向量空间模型。

向量空间模型(Vector Space Model， VSM)，它的主要任务是，将半结构化或者非结构化的文本表示成数学上可以处理的形式，这里是将文本表示成空间中

的一组向量.其基本思想是，将每个文档映射为向量空间中的一个点，它的模型如下。

任意一个文本文档 ， $t_i \in T$ ，处理后可以表示成公式的形式。

$$t_i = \left(w_{1t}, w_{2t}, \cdots, w_{nt}\right) \tag{25}$$

其中， w_{rt} 是第 1 个词在文档中的权重。通常需要构造合适的评价函数来确定词的权重，构造一个好的评价函数应该是能够最大限度的区分不同文档。以下是本文针对附件 3 进行的文本向量化得到的部分结果如下图所示：

```
(4326, 40654)
------------------------------
  (0, 34174)    0.16408359376356635
  (0, 33997)    0.1564183704479164
  (0, 33676)    0.3281671875271327
  (0, 31993)    0.2673150906892968
  (0, 31700)    0.1564183704479164
  (0, 31033)    0.1247722363370644
  (0, 30200)    0.10075443818029896
  (0, 26947)    0.03644454264019895
  (0, 25962)    0.06464270261453277
  (0, 24222)    0.08975334706577004
  (0, 23724)    0.0587792960220619
  (0, 23687)    0.12055375953581442
  (0, 23509)    0.055302565731664755
  (0, 22115)    0.09365022834115946
  (0, 21542)    0.10892381848808927
  (0, 19333)    0.06942182314221632
  (0, 18898)    0.16408359376356635
  (0, 18214)    0.16408359376356635
  (0, 18139)    0.06649861026675474
  (0, 17882)    0.1564183704479164
  (0, 16945)    0.11710701302141449
  (0, 16914)    0.14676133115348236
  (0, 16706)    0.11166845052823042
  (0, 15435)    0.3281671875271327
  (0, 13551)    0.09660241053991724
```

图 14 部分文本向量图

1.TF-IDF 函数

用 VSM 进行向量化表示过程中，需要构造计算词权重的评价函数，常用的函数有频度函数和 TF-IDF 函数。

频度函数主要计算词在文档中出现的次数，如果一个词在文档中频繁出现，那么就说明这个词对文档意思的表达至关重要，贡献较大，反之说明这个词不重

要，对文档意思的表达无关紧要，这种方法实现简单，但是忽略了那些经常出现但对文档意思表达不重要的词，用这种函数表示后的效果并不怎么理想。

TF-IDF 函数是目前非常流行计算权重的评价函数，它的基本思想是：如果一个词在一文档中出现的次数多，那么这个词在该文档中的重要度就高，但如果这个词同时在其他文档中出现的次数也很多，那么就认为这个词的区分能力(即区分某一文档和其他文档)较弱。若果这个词在其他文档中出现的次数较少，那么这个词具有很好的区分能力。

主要的计算步骤：

Step1，计算词频。

词频（*TF*)=某一类中词条w出现的次数

考虑到文章有长短，为了便于比较不同文章，进行“词频”的标准化。

$$TF_{i,j}=\frac{n_{i,j}}{\sum_k n_{k,j}} \tag{26}$$

即

$$TF_w=\frac{\text{在某一类中词条}w\text{出现的次数}}{\text{该类中所有的词条数目}} \tag{27}$$

其中 $n_{i,j}$ 是该词在文件 d_j 中出现的次数，分母则是文件 d_j 中所有词汇出现的次数总和；

Step2，计算逆文件频率。

逆向文件频率（IDF）某一特定词语的 IDF，可以由总文件数目除以包含该词语的文件的数目，再将得到的商取对数得到。如果包含词条 t_i 的文件越少，IDF 越大，则说明词条具有很好的类别区分能力。

$$idf_i=\log\frac{|D|}{|\{j:t_i\in d_j\}|} \tag{28}$$

即

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

其中，$|D|$ 是语料库总文档。$|\{j:t_i\in d_j\}|$ 表示包含词语 t_i 的文件数目。如果该词语不在语料库，就会导致分母为 0，因此一般情况下使用 $|\{j:t_i\in d_j\}|+1$

Step3，计算 TF-IDF.

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频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤常见的词语，保留重要的词语。

$$TF-IDF=TF*IDF \tag{29}$$

在进行完上述一些列处理后，原本的中文文本数据已经被处理成数学上可以处理的形式了，后续的聚类分析已经可以进行。但是，有一个问题却对后续的分析以及可视化展示有很大影响。分析发现，处理后的文档生成的向量维数比较大，有的甚至有三十多个分量，这不仅耗费存储空间，还给后面的计算带来了影响，影响计算的效率。要对一个公司所属行业进行判定，两个贡献最大的词语就足以表示。对数据进行降维是接下来至关重要的一步处理，这里我们采用主成分分析法对数据进行降维。

2.主成分分析法(Principal Component Analysis， PCA)降维

降维是处理数据很重要的步骤，降维的方法和结果直接影响到后续的计算结果。一个好的降维方法既要减少无关或者关系轻微的变量，又不能盲目减少指标损失信息。主成分分析法研究的就是怎样以丢失最少的信息将原有变量进行浓缩转换，它在这类降维中有很好的效果。

主成分分析是一种统计方法，它是将一组可能 具有相关性的相关变量，通过正交变换，转换成一组线性无关的变量来实现降维目的。例如，在三维空间中寻找一个二维直角坐标系，将三维空间中的点在这个二维直角坐标系中进行投影，就可以得到三维的点在二维空间中的表示，数据由三维降为二维。它的数学模型是，将原有的 p 个相关变量 x 标准化后进行线性组合，转换成另一组不相关的变量 y ，的方法如公式

$$\begin{cases} y_1 = u_{11}x_1 + u_{12}x_2 + u_{13}x_3 + \cdots + u_{1p}x_p \\ y_2 = u_{21}x_1 + u_{22}x_2 + u_{23}x_3 + \cdots + u_{2p}x_p \\ y_3 = u_{31}x_1 + u_{32}x_2 + u_{33}x_3 + \cdots + u_{3p}x_p \\ \cdots \\ \mathrm{y}_p = u_{p1}x_1 + u_{p2}x_2 + u_{p3}x_3 + \cdots + u_{pp}x_p \end{cases} \qquad (30)$$

（1） y_i 与 $y_j\ (i \neq j, j = 1,2,\cdots,p)$ 相互独立。

（2） y_1 是 $x_1, x_2, \cdots, x_p$ 的一切线性组合(系数满足上述方程组)中方差最大的；y_2 为是与 y_2 不相关的 $x_1, x_2, \cdots, x_p$ 的一切线性组合中方差最大的； y_p 是与 $y_1, y_2, \cdots, y_{p-1}$ 都不相关的 $x_1, x_2, \cdots, x_p$ 的一切线性组合中方差最小的。

用以上模型确定的变量 $y_1, y_2, \cdots, y_{p-1}$ 依次称为原有变量系 $x_1, x_2, \cdots, x_p$ 的第 $1,2,\cdots,p$ 个主成分。其中， y_1 火在总方差中所占比例最大，综合原有变量 $x_1, x_2, \cdots, x_p$ 的能力最强，其余 $y_1, y_2, \cdots, y_{p-1}$ 在总方差中所占比例依次递减。

主成分分析实现降维的过程主要有三步：

（1）构造并计算样本的协方差矩阵；

（2）对角化协方差矩阵；

（3）选出绝对值最大的特征值对应的特征向量为转换矩阵，进行降维。

从以上数学模型和实现过程可以看出，应用主成分分析法进行降维后，只需要选取少量的新变量就可以反映原来的绝大部分信息，在这里，我们仅取前两个方差最大的主成分进行后续分析。

文本聚类的核心思想是计算两个文本之间的相似度，将相似程度高的文本聚为一类。转化为数学描述是将空间中距离最近的两个向量归为一类。在文本聚类过程中，文本相似度的计算通常有以下几种方法。

欧几里得距离，描述的是两个点在欧几里得空间中的实际距离。见公式

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)} \tag{31}$$

欧几里得距离对文本中的词序有要求，这样很容易导致原本语义相近的两个词因为顺序不同使得距离很远。这样并不是一个很好的计算距离的方法。

最常用的计算相似度的方法是余弦相似度。判定余弦距离是否接近，是通过判断空间中两个向量的方向是否一致，如果两个向量的方向一致，那么这两个向量的夹角就越接近零，两个向量就越接近。

余弦相似是指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。当两个文本向量夹角余弦等于 1 时，这两个文本完全重复：当夹角的余弦 值接近于 1 时，两个文本相似；夹角的余弦越小，两个文本越不相关。公式如下：

$$\cos\theta=\frac{\sum_{i=1}^{n}(\mathrm{X}_i\times\mathrm{Y}_i)}{\sqrt{\sum_{i=1}^{n}(\mathrm{X}_i)^2}\times\sqrt{\sum_{i=1}^{n}(\mathrm{Y}_i)^2}} \tag{32}$$

简化

$$\mathrm{sim}(\mathrm{A},\mathrm{B})=\cos\theta=\frac{A\times B}{\|A\|*\|B\|} \tag{33}$$

通过计算模型公式可以明确的求出余弦相似度的值。那么本文我们是通过如下例子实现算法的：

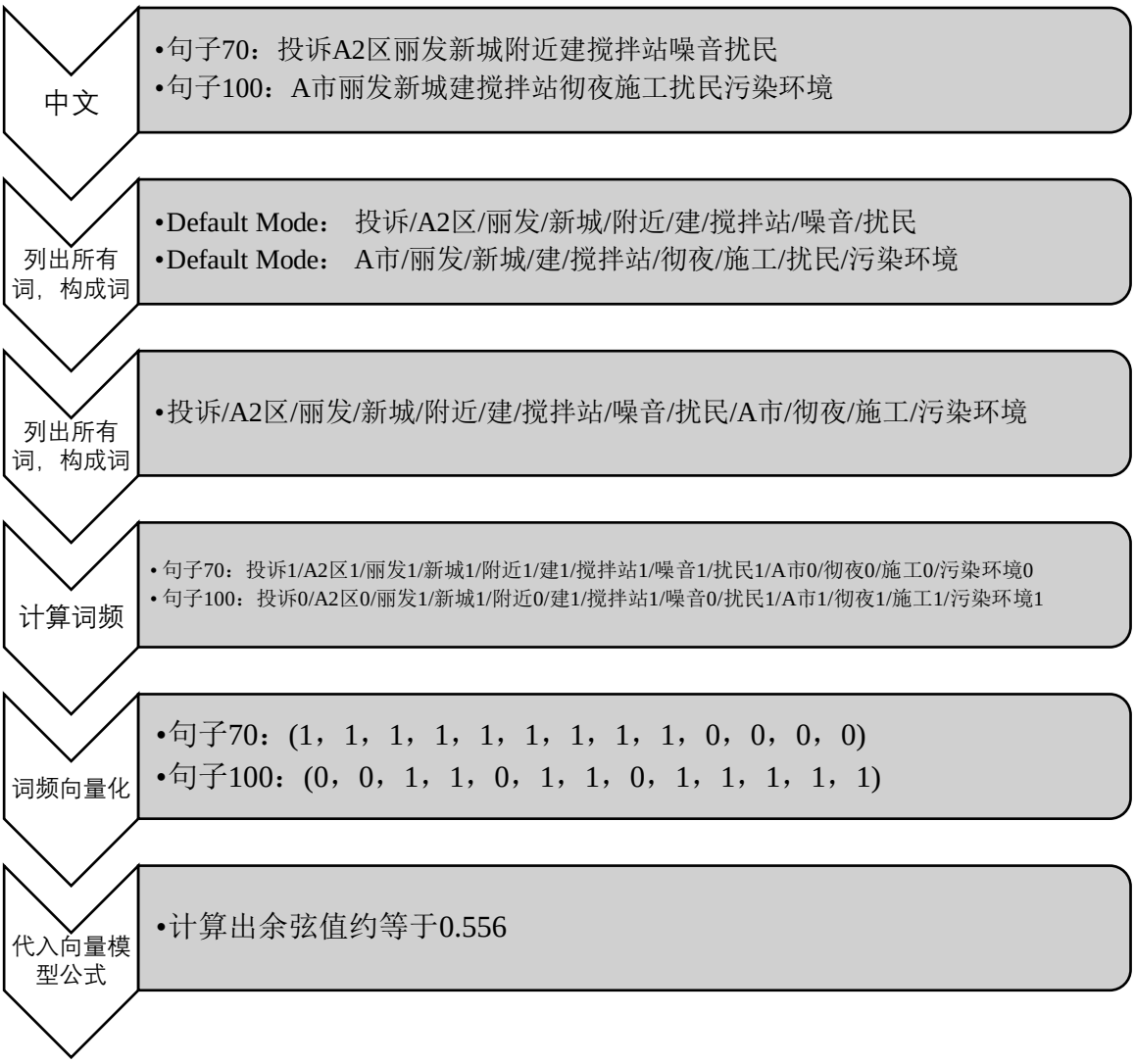

图 15 余弦相似算法图

由图 15 可知，两个句子的相似度计算的步骤是：

（1）通过中文分词，把完整的句子根据分词算法分为独立的词集合。

（2）求出两个词集合的并集(词包)。

（3）计算各自词集的词频并把词频向量化。

（4）带入向量计算模型就可以求出文本相似度。

5.2.2 文本聚类模型

目前机器学习中关于聚类的算法有很多，基于距离的文本聚类比较常用的方法大致有 2 种：划分法和层次法。

1.层次聚类（HC）

层次聚类法的核心思想是不断合并距离最近的两个类簇，直到达到聚类目的或者达到预设的条件为止。

层次聚类的算法如下：

(1)将数据集中的每个样本点当作一个类别。

(2)计算所有样本点之间的两两距离，并从中挑选出最小距离的两个点构成一个簇。

(3)继续计算剩余样本点之间的两两距离和点与簇之间的距离，然后将最小距离的点或簇合并到一起。

(4)重复步骤(2)和(3)，直到满足聚类的个数或其他设定的条件，便结束算法的运行。

在划分法中最典型的算法是 K 均值聚类算法。

K-Means 聚类算法

K-Means 算法过程如下：

（1）从 N 个样本数据中随机选取 K 个对象作为初始的聚类质心。

（2）分别计算每个样本到各个聚类中心的距离，将对象分配到距离最近 的聚类中。

（3）所有对象分配完成之后，重新计算 K 个聚类的质心。

（4）与前一次的 K 个聚类中心比较，如果发生变化，重复过程（2），否 则转到过程。

（5）当质心不再发生变化时，停止聚类过程，并输出聚类结果。

在这个过程中有两个关键步骤，一是按最近准则分配数据对象时，计算数据对象距离最近中心点的过程中，为每个数据对象“贴上”标签。也就是说，为了方便联系数据对象与距离其最近的中心点，我们为每个数据对象分配一个标签。例如，距离初始中心点 A 最近的数据对象，我们把标签 A 分配给该数据对象，距离初始中心点 B 最近的数据对象，把标签 B 分配给该数据对象。如此循环，直到所有的点都被分配完毕为止。二是计算每个类的均值，更新聚类中心。在更新聚类中心的过程中，聚类中心或多或少都会发生一些改变，影响到每一个数据对象的均值，那么聚类中心选择停止的条件就是当聚类中心不再改变时或者仅发生特别微小的变动时，可以认为聚类中心已经不再改变，聚类过程完成，算法停止运行。采用的目标函数形式为平方误差准则函数见式。

$$J(\mathrm{c}_1,\mathrm{c}_2,\cdots,\mathrm{c}_k)=\sum_{j=1}^{k}\sum_{i}^{n_j}(\mathrm{x}_i-\mathrm{c}_j)^2 \qquad (34)$$

其中，c_j 表示第 j 个簇的簇中心，x_i 属于第 j 个簇的样本 i，n_j 表示第 j 个簇的样本总量。对于该目标函数而言，c_j 数未知的参数；要想求得目标的最小值，得先知道参数 c_j 的值。由于目标函数J为一个凸函数，因此可以通过求导的方式获取合理的参数 c 的值。

步骤一：对目标函数求偏导

$$\frac{\partial J}{\partial c_j}=\sum_{j=1}^{k}\sum_{i=1}^{n_j}\frac{(\mathrm{x}_i-\mathrm{c}_j)^2}{\partial c_j}=\sum_{i=1}^{n_j}\frac{(\mathrm{x}_i-\mathrm{c}_j)}{\partial c_j}=\sum_{i=1}^{n_j}-2(\mathrm{x}_i-\mathrm{c}_j) \qquad (35)$$

由于仅对目标函数中的第 j 个簇中心 c_j 偏导，因此其他簇的离差平方和的导

数均为 0，进而只保留第 j 个簇的离差平方和的导函数。

步骤二：令导函数为 0

$$\sum_{i=1}^{n_j} -2(\mathrm{x}_i - \mathrm{c}_j) = 0 \tag{36}$$

$$n_j c_j - \sum_{i=1}^{n_j} x_i = 0 \tag{37}$$

$$c_j = \frac{\sum_{i=1}^{n_j} x_i}{n_j} = u_j \tag{38}$$

在做 K-Means 聚类时，首先要确定 k 值，k 值的确定有本文有两种方法，轮廓系数和手肘法，廓系数该方法的核心指标是轮廓系数（Silhouette Coefficient），某个样本点Xi的轮廓系数定义如下

$$S = \frac{b-a}{max(a,b)} \tag{39}$$

其中 a 是Xi和同簇的其他样本的平均距离，称为凝聚度。b 是 X_i 和最近簇中所有样本的平均距离，称之为分离度。最近簇的定义如下

$$C_j = argminC_k \frac{1}{n} \sum_{p \in Ck} | p - Xi |^2 \tag{40}$$

其中 p 是簇 C_k 中的样本， 也就是说，计算出 X_i 到所有簇的平均距离之后，选取最小的作为 b 即可。

但是在实践的时候发现手肘图的最佳值和轮廓系数的最佳值是不一样的，轮廓系数的最佳值小于手肘图的最佳值，原因我猜测是轮廓系数考虑了分离度 b，也就是样本与最近簇中所有样本的平均距离。

从定义上看，轮廓系数大，不一定是凝聚度 a（样本与同簇的其他样本的平均距离）小，而可能是 b 和 a 都很大的情况下 b 相对 a 大得多，这么一来， a 是有可能取得比较大的。a 一大，样本与同簇的其他样本的平均距离就大，簇的紧凑程度就弱，那么簇内样本离质心的距离也大，从而导致 SSE 较大。所以，虽然轮廓系数引入了分离度 b 而限制了聚类划分的程度，但是同样会引来最优结果的 SSE 比较大的问题,得出本文最好用手肘法。

手肘法的评价 K 值好坏的标准是 SSE

$$SSE = \sum_{p \in Ci} | p - mi |^2 \tag{41}$$

其中 C_i 代表第 i 个簇， p 是簇 C_i 里的样本点， m_i 是簇的质心。

手肘法的核心思想是：随着聚类数 k 的增大，样本划分会更加精细，每个簇的聚合程度会逐渐提高，那么误差平方和 SSE 自然会逐渐变小。并且，当 k 小于最佳聚类数时，由于 k 的增大会大幅增加每个簇的聚合程度，故 SSE 的下降幅度

会很大，而当k到达最佳聚类数时，再增加k所得到的聚合程度回报会迅速变小，所以 SSE 的下降幅度会骤减，然后随着k值的继续增大而趋于平缓，也就是说 SSE 和k的关系图是一个手肘的形状，而这个肘部对应的k值就是数据的最佳聚类数。根据算法确定$k=100$时时最佳聚类数，如图 16 所示

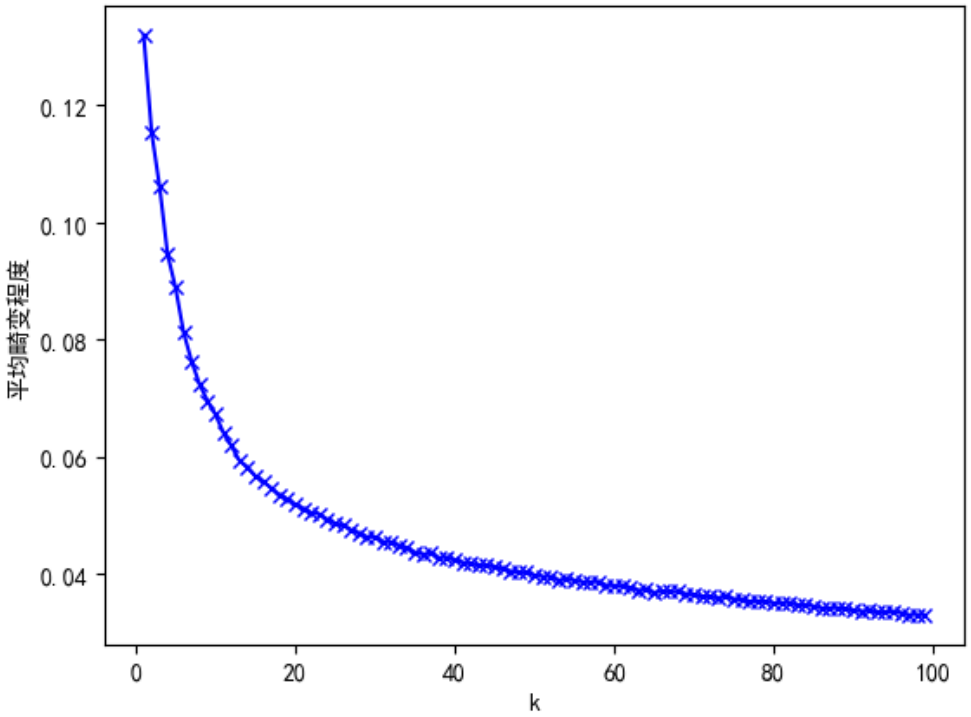

图 16 用肘部法确定最佳的k值

结果显示出当附件 3 聚成 100 类时，ID=10 时，该热点问题被反映次数最多为 110 次，具体详情请见表 1-热点问题和 2-热点问题明细表，得出结论：热点问题中：A 市地区夜晚施工，造成扰民现象被群众反映最多。

5.2.3 模型比较评价

层次聚类这个过程中会构造出一 棵聚类树，树中包含了簇的层次信息。层次聚类生成的聚类属可以清晰的展示出类之间的关系以及类簇合并的过程，类的划分准确度较高，而且可以根据自身需要决定最终划分为几类，决定循环终止的条件。但是层次聚类也有自身无法规避的劣势。在每次合并之前，需要计算比较所有簇之间的距离并选出距离最近的两个簇合并，算法的时间复杂度是$o(n^3)$，这样将导致算法整体计算速度非常慢，不适合数据量较大的情况。本文在初步构思时采用了层次算法进行聚类，用 Java 编写聚类算法处理数据时，仅仅是 500 条数据，也需要将近两分钟才能得到聚类结果，数据量较大时，从时间成本来说，这一算法 并不合适。因而在后期数据处理过程中，本文采用了 K 均值算法来进行聚类。

从算法本身来看，初始聚类中心的选取和分成几类的选择对聚类结果的影响

比较明显，但是算法的执行速度比较快，数据量较大时，K 均值算法的时间复杂度是o(hkn)，t 是迭代次数，k 是聚类数目，n 是数据量，这在聚类速度上有明显的优势。本次研究中，由于获得的数据量较大，对聚类速度有一定要求，所以在文本聚类过程中，采用 K 均值聚类算法更为合理。最终结果也表明，该算法在本此研究中取得了很好的聚类效果。

5.3 答复意见评价

5.3.1 模型准备

针对问题 3 答复意见的评价，我们采取对附件 4 文本分词、去除停用词、文本向量化、PCA 降维等进行预处理，此步骤已在本文预处理模块进行详细阐述，这里就不在声明。

5.3.2 相关性分析

本文采取从答复的相关性进行分析判断两个变量是否具有线性相关关系最直观的方法是直接绘制散点图，看 变量之间是否符合某个变化规律。当需要同时考察多个变量间的相关关系时， 一一绘制他们间的简单散点图是比较麻烦的。此时可以利用散点矩阵图同时绘 制各变量间的散点图，从而快速发现多个变量间的主要相关性，这在进行多元 线性回归时显得尤为重要。 相关分析是研究现象之间是否存在某种依存关系，并对具体有依存关系的现象探讨其相关方向以及相关程度，是研究随机变量之间的相关关系的一种统 计方法。 为了更加准确地描述变量之间的线性相关程度，通过计算相关系数来进行 相关分析，在二元变量的相关分析过程中，比较常用的有 Pearson 相关系数、 Spearman 秩相关系数和判定系数。 Pearson 相关系数一般用于分析两个连续变 量之间的关系，要求连续变量的取值服从正态分布。不服从正态分布的变量、 分类或等级变量之间的关联性可采用 Spearman 秩相关系数（也称等级相关系 数）来描述。 相关系数可以用来描述定量变量之间的关系。 相关系数与相关程度之间的关系如表所示。

5.3.3 答复意见评价指标

相关分析函数有：

DataFrame.corr()

Series.corr(other)

如果由 DataFrame 调用 corr 方法，那么将会计算每列两两之间的相似度。如果由序列调用 corr 方法，那么只是计算该序列与传入的序列之间的相关度。DataFrame 调用返回值：返回 DataFrame; Series 调用返回值：返回一个数值型，大小为相关度。以下便是相关性分析评价结果请见表答复相关性，从表中可见答

复意见相关性较低。

6 参考文献

[1] 刘顺祥. 从零开始学 Python 数据分析与挖掘[M].北京:清华大学出版社,2018,108-343

[2] 姜启源，谢金星，叶俊，数学模型（第五版），北京：高等教育出版社，2018

[3] 王学民. 应用多元统计分析[M]. 上海:上海财经大学出版社,2017,258-294.

[4] 李宝仁. 计量经济学[M]. 北京:机械工业出版社,2015,56-68

[5] 司守奎. 数学建模算法与应用. 北京:国防工业出版社,2011，569-575

[6] 余本国.基本 Python 的大数据分析基础及实战.中国水利水电出版社

相关系数 $\|r\|$ 取值范围	相关程度
$0 \le \|r\| < 0.3$	低度相关
$0.3 \le \|r\| < 0.8$	中度相关
$0.8 \le \|r\| \le 1$	高度相关

附录

```
from sklearn.decomposition import PCA
import nltk
from nltk.cluster.util import cosine_distance
from nltk.cluster. kmeans import KMeansClusterer
from pandas import DataFrame
import networkx as nx
from sklearn import metrics
from pandas import DataFrame
import nltk
from nltk.cluster.kmeans import KMeansClusterer
from sklearn.metrics import confusion_matrix
import pandas as pd
```

```
from pandas import read_excel
import matplotlib
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
import jieba.posseg as pseg
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from pandas import Series

def CutWord(df):
    tdf = df.copy(deep = True)
    print("开始分词")
    cuttheme = []
    details = []
    for i in range(len(tdf)):
        print("正在进行分词"+str(i))
        tdf.iloc[i,2] = tdf.iloc[i,2].replace("\r\n", "")#删除换行
        tdf.iloc[i,2] = tdf.iloc[i,2].replace(" ", "")#删除空格
        tdf.iloc[i,4] = tdf.iloc[i,4].replace("\r\n", "")#删除换行
        tdf.iloc[i,4] = tdf.iloc[i,4].replace(" ", "")#删除空格
        cuttheme = jieba.cut(tdf.iloc[i,2])
        details = jieba.cut(tdf.iloc[i,4])
        tdf.iloc[i,2] = ""
        tdf.iloc[i,4] = ""
        for every in cuttheme:
            if every not in stopword:  #去掉停用词
                tdf.iloc[i,2] = (tdf.iloc[i,2] + " " + every.strip())
        for every in details:
            if every not in stopword:
                tdf.iloc[i,4] = (tdf.iloc[i,4] + " " + every.strip())
    print("分词结束")
return(tdf)


def Accuracy(clf,X_test,y_test):
    count = 0
```

```
        x = list(X_test)
        y = list(y_test)
        y_pred=Series()
        for i in range(len(X_test)):
            format_sec="".join(x[i])
            category=clf.predict(count_vect.transform([format_sec]))
            y_pred = y_pred.append(Series(category[0]))
            if category[0]==y[i]:
                count = count + 1
        print(count/(len(x)))
        return(list(y_pred))

    X_train, X_test, y_train, y_test = train_test_split(df_x, cut_df["一级标签"], random_state = 0)
    count_vect = CountVectorizer()
    X_train_counts = count_vect.fit_transform(X_train)
    tfidf_transformer = TfidfTransformer()
    X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

    df_test = pd.DataFrame(list(zip(X_test, y_test)))
    df_cx = df_test[df_test[1].str.contains("城乡建设")]
    df_ld = df_test[df_test[1].str.contains("劳动和社会保障")]
    df_jy = df_test[df_test[1].str.contains("教育文体")]
    df_sm = df_test[df_test[1].str.contains("商贸旅游")]
    df_hj = df_test[df_test[1].str.contains("环境保护")]
    df_ws = df_test[df_test[1].str.contains("卫生计生")]
    df_jt = df_test[df_test[1].str.contains("交通运输")]

    t = (df_cx,df_ld,df_jy,df_sm,df_hj,df_ws,df_jt)

    ##朴素贝叶斯分类
    print("贝叶斯分类：")
    clf_NBM = MultinomialNB().fit(X_train_tfidf, y_train)
    for i in range(len(t)):
        print(str(list(t[i][1])[0])+"分类准确率：")
        Accuracy(clf_NBM,t[i][0],t[i][1])

    ##支持向量机
    print("支持向量机")
    clf_svm = SVC(kernel='linear')
    clf_svm.fit(X_train_tfidf,y_train)
    for i in range(len(t)):
```

```
        print(str(list(t[i][1])[0])+"分类准确率：")
        Accuracy(clf_svm,t[i][0],t[i][1])

    ##KNN
    print("KNN 分类")
    knn = KNeighborsClassifier()
    knn.fit(X_train_tfidf,y_train)
    for i in range(len(t)):
        print(str(list(t[i][1])[0])+"分类准确率：")
        Accuracy(knn,t[i][0],t[i][1])
    res = confusion_matrix(y_test, Accuracy(knn,X_test,y_test),labels = ["城乡建设", "劳动和社会保障", "教育文体","商贸旅游","环境保护","卫生计生","交通运输"])
    print(res)

    res = confusion_matrix(y_test, Accuracy(clf_NBM,X_test,y_test),labels = ["城乡建设", "劳动和社会保障", "教育文体","商贸旅游","环境保护","卫生计生","交通运输"])
    print("混淆矩阵")
    print(res)

    res = confusion_matrix(t[0][1], Accuracy(clf_NBM,t[0][0],t[0][1]),labels = ["城乡建设", "劳动和社会保障", "教育文体","商贸旅游","环境保护","卫生计生","交通运输"])
    print("混淆矩阵")
    print(res)

    tfidf = TfidfVectorizer(norm='l2', ngram_range=(1, 2))
    features = tfidf.fit_transform(cut_df_train["留言详情"])
    labels = df_train["一级分类"]
    print(features.shape)
    print('----------------------------')
    print(features)

    def CutWord(df):
        tdf = df.copy(deep = True)
        print("开始分词")
        cuttheme = []
        details = []
        for i in range(len(tdf)):
            print("正在进行分词"+str(i))
            tdf.iloc[i,2] = tdf.iloc[i,2].replace("\r\n", "")#删除换行
```

```
            tdf.iloc[i,2] = tdf.iloc[i,2].replace(" ", "")#删除空格
            tdf.iloc[i,4] = tdf.iloc[i,4].replace("\r\n", "")#删除换行
            tdf.iloc[i,4] = tdf.iloc[i,4].replace(" ", "")#删除空格
            tdf.iloc[i,5] = tdf.iloc[i,5].replace("\r\n", "")#删除换行
            tdf.iloc[i,5] = tdf.iloc[i,5].replace(" ", "")#删除空格
            cuttheme = jieba.cut(tdf.iloc[i,2])
            details = jieba.cut(tdf.iloc[i,4])
            reply = jieba.cut(tdf.iloc[i,5])
            tdf.iloc[i,2] = ""
            tdf.iloc[i,4] = ""
            tdf.iloc[i,5] = ""
            for every in cuttheme:
                if every not in stopword:  #去掉停用词
                    tdf.iloc[i,2] = (tdf.iloc[i,2] + " " + every.strip())
            for every in details:
                if every not in stopword:
                    tdf.iloc[i,4] = (tdf.iloc[i,4] + " " + every.strip())
            for every in reply:
                if every not in stopword:
                    tdf.iloc[i,5] = (tdf.iloc[i,5] + " " + every.strip())
        print("分词结束")
        return(tdf)

    df3=pd.read_excel(r"C:\Users\ASUS\Desktop\泰迪杯\C 题全部数据\附件4.xlsx")
    df3.head()
    cut_df3 = CutWord(df3)
    a = cut_df3["留言主题"]+cut_df3["留言详情"]
    b = cut_df3["答复意见"]

    count_vect = CountVectorizer()
    a_counts = count_vect.fit_transform(a)
    tfidf_transformer = TfidfTransformer()
    a_tfidf = tfidf_transformer.fit_transform(a_counts)

    count_vect = CountVectorizer()
    b_counts = count_vect.fit_transform(b)
    tfidf_transformer = TfidfTransformer()
    b_tfidf = tfidf_transformer.fit_transform(b_counts)

    a_dtm = a_tfidf.toarray()
    b_dtm = b_tfidf.toarray()
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 100)
pca.fit(a_dtm)
coord_a = pca.fit_transform(a_dtm)

pca = PCA(n_components = 100)
pca.fit(b_dtm)
coord_b = pca.fit_transform(b_dtm)


c = []
for i in range(len(DataFrame(coord_a))):
    c.append(DataFrame(coord_a).iloc[i].corr(DataFrame(coord_b).iloc[i]))


c = DataFrame(c)
c.to_excel("答复相关性.xlsx", index=True)

DataFrame(coord_a).iloc[0].corr(DataFrame(coord_b).iloc[0])
```