“智慧政务”中的文本挖掘应用

摘要

随着网络问政平台的发展，政府可以更好地了解民意、汇聚民智、凝聚民气。亿万条留言的划分和热点整理的相关部门的工作带来了极大挑战。这就需要建立基于自然语言处理技术的智慧政务系统来更智能、更高效地为政府工作带来便捷。

针对问题一，我们在对数据预处理后，首先建立了向量空间（VSM）模型使文本数据向量化，提取特征值后，对词频矩阵划分训练集和测试集，建立随机森林模型和支持向量机模型进行训练，通过计算得到F-score。利用R语言编程对数据进行了清洗，清除数据噪声；利用TF-IDF方法求解VSM模型，得到向量化数据；使用文档频数的特征选择方法从所有特征中提取243个文本特征；将词频矩阵的70%划为训练集，剩下的30%为测试集，训练随机森林模型和支持向量机模型，用R软件编程求解，得到两个模型的F1值，通过比较F1值的大小，认为随机森林模型方法准确度更高。

针对问题二，我们将问题一的向量空间模型与LDA模型相结合，将文本向量化；为了寻找相似的文本，我们利用K-means文本聚类的方法自动地将文本分为不同主题，按照不同主题将留言分类并以主题数量由高到低重新排序，并得到前5个热点主题，最后统计出各主题的时间跨度并分析合理性，得到最终热点问题分类。

针对问题三，我们首先定义了留言答复意见数量化的指标，根据答复意见是否符合该指标计算其得分，最后由质量指标得分判断答复意见的质量等级。通过人为判断得分，得到训练集，进而通过随机森林模型判断其他测试文本的等级。

关键词：文本分词、随机森林分类、LDA模型、k-means聚类

目录

一、问题重述 4
1.1 问题一 4
1.2 问题二 4
1.3 问题三 4
二、问题分析 4
2.1 问题一 4
2.2 问题二 5
三、模型假设 5
四、模型建立 5
4.1 问题一模型建立 5
4.1.1 数据预处理 5
4.1.2 向量空间模型 6
4.1.3 文本特征选择 6
4.1.4 分类模型 7
4.1.4.1 随机森林 7
4.1.4.2 支持向量机 7
4.2 问题二模型建立 8
4.2.1 清洗文本 9
4.2.2 文本相似度 9
4.2.3 文本聚类 10
4.3 问题三模型建立 10
4.3.1 答复意见量化处理 10
4.3.2 利用随机森林模型进行测试 10
五、模型求解 10
5.1 模型一求解 10
5.1.1 数据预处理 10
5.1.2 分词及特征选取 11
5.1.3 分类模型 12

5.1.3.1 随机森林模型 12
5.1.3.2 支持向量机模型 14
5.2 问题二模型求解 14
5.2.1 清洗文本 14
5.2.2 文本聚类 14
5.3 问题三求解 15
5.3.1 答复意见量化处理 15
5.3.2 随机森林模型测试 16
六、模型检验 17
6.1 随机森林模型 17
6.1.1 灵敏度和特异度 17
七、模型评价 17
八、参考文献 18
九、附录 18

一、问题重述

1.1 问题一

在处理网络问政平台的群众留言时，工作人员要先按照附件 1 提供的内容分类三级标签体系对留言进行分类，后将群众留言分派至相应的职能部门处理。由于现在大部分电子政务系统仍需人工经验处理，所以存在着工作量大、效率低，差错率高的问题。根据群众的留言书局可以建立关于留言内容的一级标签分类模型。模型优劣常使用 F-Score 对分类方法进行评价：

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

其中 Pi 为第 i 类的查准率，Ri 为第 i 类的查全率。

1.2 问题二

热点问题的定义为某一时段内群众集中反映的某些问题。为了使有关部门及时发现问题并解决问题，需要及时发现热点问题。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，列出排名前 5 的热点问题，同时需要给出相应热点问题对应的留言信息。

1.3 问题三

针对附件 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。

二、问题分析

2.1 问题一

问题一需要得到关于一级标签的分类模型，附件 2 的一级标签为“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”共 7

类。我们首先要先对文本进行预处理，删掉无关信息，减少数据干扰；构建向量空间（VSM）模型，使文档数据化，将词语出现频率表示维度的长，通过 TF-IDF 计算将文本转化为向量；文档数据化后，进一步我们采用文档频数（DF）的特征选择方法提取文本特征。

得到词频矩阵后，我们采用 RF（随机森林）模型和 SVM（支持向量机）模型对我们划分的训练集和测试集进行训练和测试，并得到模型的准确率。计算查准率和查全率并得到两个模型下的 F1 值，比较得到两个模型的优劣。

2.2 问题二

问题二要求寻找某一时段内群众集中反映的热点问题，即分析文本需要从两个角度出发，我们考虑以舆论问题的热度为主要角度进行文本的主题分类，再根据主题汇总之后分析每个主题的时间跨度情况。

由此出发，我们仍旧将文本向量化并将 TF-IDF 与 LDA 模型结合寻找相似的文本，用 K-means 文本聚类的方法自动地将文本分为不同主题。按照不同主题将留言分类并以主题数量由高到低重新排序，最后统计出各主题的时间跨度并分析合理性，得到最终热点问题分类。

三、模型假设

1、假设每一条群众留言都是真实可信的数据，其反映的问题与地点是准确的。
2、假设文本清洗使用停用词较为完备，分词函数的效果好。
3、各个分词完全独立，不存在词意重复。

四、模型建立

4.1 问题一模型建立

4.1.1 数据预处理

数据预处理是文本分类的首要一步。由于我们处理的文本文档中有很多转义字符、标点和无意义的词，例如“今天”，“我”等，所以需要先对文本文档进行数据清洗，减少数据噪声。

4.1.2 向量空间模型

建立向量空间模型（SVM）使文本数据转换成数据矩阵类型，使计算机能够处理。通过 TF-IDF 方法：

$$\mathrm{W(t,d)} = \frac{tf(t,d) \times \log(\frac{N}{n_i} + 0.01)}{\sqrt{\sum_{t \in d}[tf(t,d) \times \log(\frac{N}{n_i} + 0.01)]}}$$

（其中 W(t,d)为词 t 在文本 d 中的权重，tf(t,d)为 t 在文本 d 中的词频，N 为训练样本总数，n_i为训练文本集中出现 t 的文本数量）

将每一篇文档都映射为一个向量（W1,W2,W3…W9210）,其中 W_i 是第 i 个特征值的权重，我们用相对词频作为权重。

4.1.3 文本特征选择

我们选择了权重大于等于 20 的词作为特征词，由于文本中频率大于 20 的特征词个数过多，所以我们对其进行特征词提取。我们选择互信息特征提取法对我们找到的特征词进行提取。特征词和类别的互信息越大，说明特征词中包含和类别相关的信息越多，根据特征词分类的准确度越高。

词在类别出现的比重：

$$\mathrm{P(t|}C_k) = \frac{1 + \sum_{i=1}^{|D|} N(t, d_i)}{|T| + \sum_{j=1}^{|T|} \sum_{i=1}^{|D|} N(t_j \cdot d_i)}$$

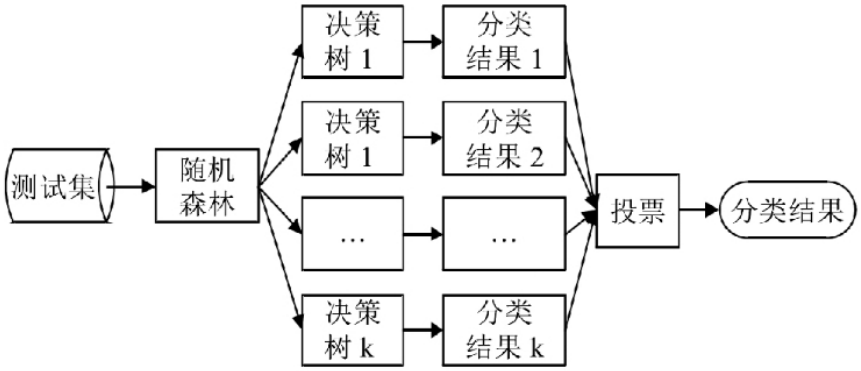
其中，$|D|$ 是该类的训练集文本数，$\sum_{j=1}^{|T|} \sum_{i=1}^{|D|} N(t_j \cdot d_i)$为该类所有词的词频和，$N(t, d_i)$为词 t 在$d_i$中的词频，$|T|$为总词数

词与类别的互信息量为 $\log\frac{P(t|C_k)}{P(t)}$,其中 P(t)为词在所有训练样本中的比重。将取值最大的前 243 个特征保留，删去其他特征，完成特征选取。

4.1.4 分类模型

4.1.4.1 随机森林

结合 bagging 算法和随机子空间，建立随机森林模型，对群众留言文本进行分类。如图所示：



随机森林由决策树组成。从所有文本 N 个中采用 bootstrap 抽样得到 N 个样本成为训练子集。从 M 个特征属性中，随机抽样 m 个作为候选特征，在决策树每个节点按照 Gini 指数进行分裂，Gini 指数越小，数据分割越彻底：

$$\mathrm{Gini(p)} = \sum_{k=1}^{K} p_k(1-p_k)$$

直到该节点所有训练样本都为同一类有。重复这种抽样 k 次，直到获得 k 个决策树，生成随机森林。由得到的随机森林分类器中的决策树共同投票来决定文本究竟属于哪一类。

设 x 为测试样本，h_i为单棵决策树，h_i为示性函数，Y 为最终分类，H 为随机森林模型，其决策公式为：

$$\mathrm{H(x)} = \arg\max_{Y} \sum_{i=1}^{k} I(h_i(x) = Y)$$

4.1.4.2 支持向量机

将样本数据映射至高维空间，并在这个高维空间中找到最大间隔超平面，该平面可以将不同类别的数据分割开来。

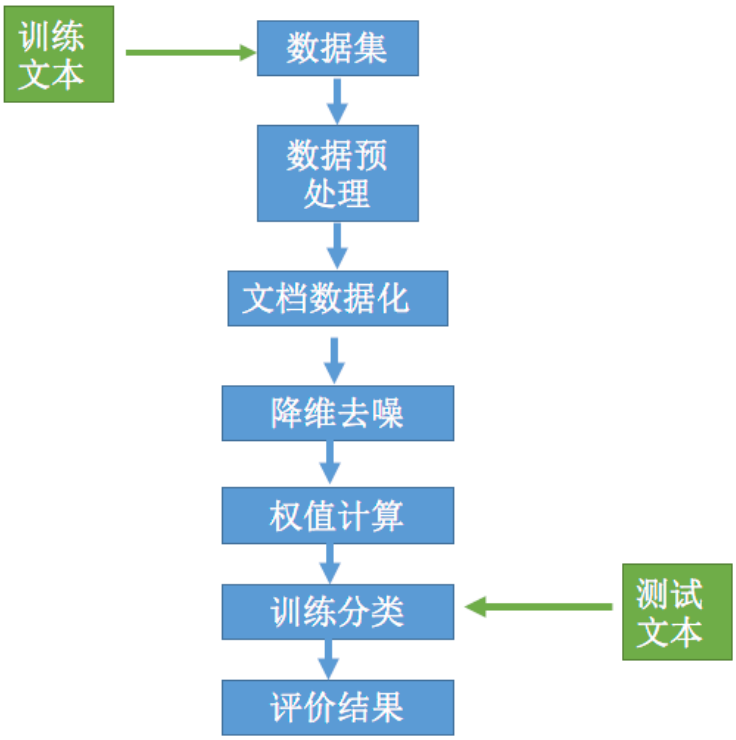
利用拉格朗日乘子法满足 KKT 条件，最终得到最优分类函数：

$$f(x) = sgn\{w^* \cdot x + b\}$$

$$sgn\{\sum_{i=1}^{k} a^* y_i (x_i \cdot x) + b^*\}$$

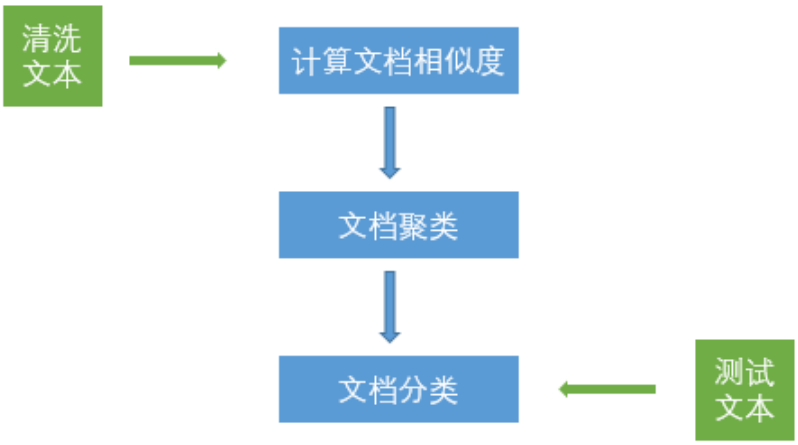
其中a^*，b^*为确定最优平面的参数。

我们最终的解题想法如下所示：



4.2 问题二模型建立

对于问题二而言，我们最重要的事情是在所有的留言评论中找出相似的文本进行聚类。总的解题算法如下图所示。



4.2.1 清洗文本

同问题一的数据处理模式相同，我们首先对文本进行了清洗，并利用 R 语言的‘tidyr’包对文本进行分词，利用“哈工大停止词典”删除文本中的停止词，并统计词频。对文本中的词频进行一个初步的观察，然后筛选必要的名词、形容词、动词等对分类问题比较有意义的词并保留，也可以与最后的计算结果相照应。

4.2.2 文本相似度

如果两文本的特征向量为

$$Vd_i = (w_{i,1} \quad w_{i,2} \quad \cdots \quad w_{i,n}),\ Vd_j = (w_{j,1} \quad w_{j,2} \quad \cdots \quad w_{j,n})$$

并且两特征向量在空间中的夹角为 θ，那么它们相似度衡量的方法主要有利用内积、Jaccard 系数、Dice 系数和余弦系数，本文中我们主要选取两个向量的夹角余弦值来衡量文本相似度：

$$S(d_i, d_j) = \cos\theta = \frac{\sum_{k=1}^{n} w_{ik} * w_{jk}}{\sqrt{(\sum_{k=1}^{n} w_{ik}^{2})(\sum_{k=1}^{n} w_{jk}^{2})}}$$

S 越大，表示两个文本之间的相似程度越高。

在本题中，我们依旧使用基于词频的 TF-IDF 模型将文本信息转化为向量信息，文本向量表征的好坏会决定最后聚类理论上能否得到最好结果。同时由于文本分词得到的数量结果过高，即使通过筛选向量空间维度也很高，所以我们仍旧先降维再计算结果，可以通过模型得到词向量间的相似度并寻找对应关系。

最后，我们使用基于 LDA 模型下的向量来寻找相似的文本，学习每个文档的主题表示以及与每个主题相关联的词语。其中寻找的算法——崩溃的吉布斯采

样如下：

- 设置需要分得的 K 个主题，浏览每个文档，并将文档中的单词随机分配给 K 中的一个；
- 为文档选择主题混合集上的 Dirichlet 概率分布；
- 给出目前主题表示的理论单词分布，将其与实际文档中的单词分布进行比较；
- 不断重复上述过程，更新主题中的单词。

4.2.3 文本聚类

计算文本相似度之后使用 K-means 文本聚类方法自动地将文本集合分组为不同地类别，保证同个类别中地文本尽可能地相似，即拥有共同的主题。

K-means 聚类通过初始化 k 个簇心点计算每个点与簇心点的距离，将每个点划分到距离该点最近的簇心中；重新计算点与簇心的距离，更新簇中心，如此迭代一定次数，或者前后两次每个点到簇中心距离的综合不超过设定的阈值则停止迭代，输出聚类结果。

4.3 问题三模型建立

4.3.1 答复意见量化处理

我们考虑从相关性、完整性、可解释性三个角度，对相关部门的答复意见的质量进行量化处理。

4.3.2 利用随机森林模型进行测试

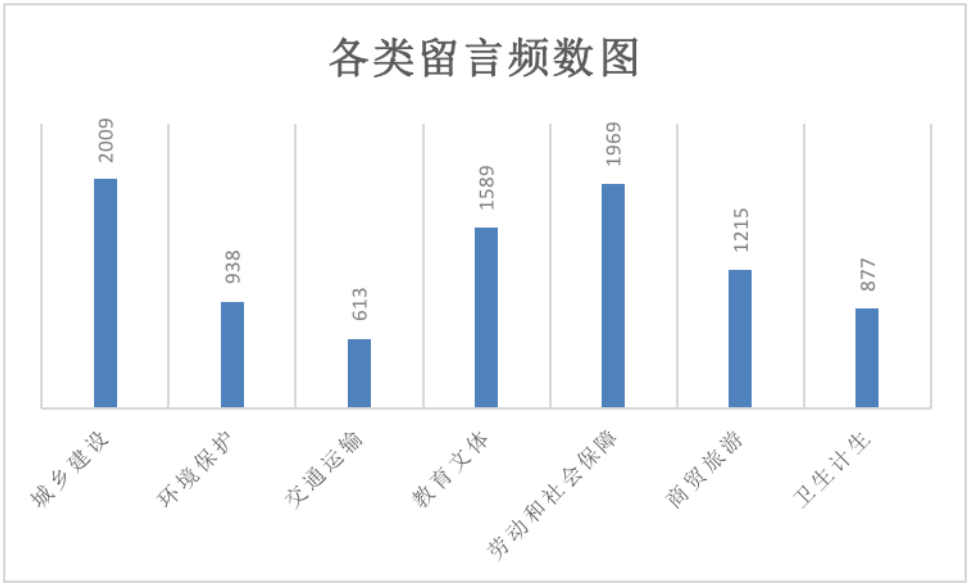
仍沿用问题一的随机森林模型对得到的结果进行训练。

五、模型求解

5.1 模型一求解

5.1.1 数据预处理

我们使用的数据为附件 2 的群众意见数据，共 9210 条留言。该文件共有 7 种类型，分别为“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”



我们将不同类别的文本按照列表形式储存，每一维对应一个分类，是由该类型文本组成的一个向量。

得到文本数据后，将其导入 R 语言软件。由于文本中含有大量的年份数字、大小写字母、多种转义字符，且文本无关信息较多。所以我们需要对文本文档做数据清洗。同时我们删去了文本长度低于 20 的数据和不相关的部分文本数据。处理后剩下 7113 条留言。

5.1.2 分词及特征选取

与英文不同，中文的每个词没有固定的分隔符。所以我们需要先对文本进行分词。我们使用 R 软件的‘jiebaR’包进行分词。由于该包内的停止词并不完全适用我们的数据，所以我们又增添了例如“每天”，“问题”等词。同时将文章中的停止词、低频词删除，同义词合并。我们选择频率大于等于 20 的词作为文本的特征词。

我们以词在文本中的频率作为特征选取的主要指标。首先计算各个特征词的互信息值的大小（其公式在 5.1.3 中已给出）并从大到小进行排序，选取互信息最大的 243 个作为特征词，这 243 个特征词是使得分类效果最好。

下表选择了 12 个特征词及其在文本中的频数（243 个特征词见附件）

特征词	频数

教师	4258
教育	4068
学校	3835
学生	2450
劳动	2351
保险	2044
医院	2031
职工	1975
退休	1579
孩子	1502
污染	1295
小学	1222

5.1.3 分类模型

5.1.3.1 随机森林模型

我们得到词频矩阵之后，将其划分成训练集和测试集用来训练随机森林模型。从词频矩阵中随机抽样 70%文本数据作为训练集，剩下的 30%文本数据为测试集。

首先对训练集进行有放回抽样，抽取次数与训练集文本个数相同，获得的样本做为训练集的子集成为新的训练集；在生成的新训练集中，随机抽取特征形成一个子集，利用 C4.5 算法用该子集进行决策树的训练，让每棵树尽可能生长不对其剪枝，每一棵树的训练集都不同，且包含重复样本；重复上述方法，直到训练出 500 棵决策树；使用 R 软件编程求解得到下列混淆矩阵：

	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	class.error
城乡建设	713	41	29	38	48	63	22	0.2526
环境保护	71	399	2	10	6	5	7	0.202

交通运输	31	4	300	3	5	21	3	0.1825
教育文体	44	6	1	856	39	23	13	0.1283
劳动和社会保障	71	4	10	35	885	18	65	0.1865
商贸旅游	105	11	22	19	21	434	18	0.2911
卫生计生	20	2	1	20	30	19	366	0.2008

该矩阵的最左侧一列为训练集的文本的真实类别，最上面一行为预测出的类别。基本上每一行的错判率均在 0.3 以下。袋外错判率（OOB）为 19.6%，我们认为模型尚可。

将未分类的测试集放入随机森林模型中，每个决策树对其进行分类投票，将投票最多的类别作为最终分类的结果。求解得到真实值和测试值的混淆矩阵（见附录）并计算每一类的查准率P_i，由于本题是多分类问题，使用查准率不同于二分类：

$$P_i = \frac{正确预测第i类文档的数目}{预测为第i类文档的总数目}$$

$\mathrm{P} = (P_1, P_2, P_3, P_4, P_5, P_6, P_7)$=(0.8019,0.7782,0.8159,0.84,0.846,0.66,0.7991)

同时计算其查全率R_i

$$R_i = \frac{正确预测第i类文档的数目}{第i类所有文档的数目}$$

$\mathrm{R} = (R_1, R_2, R_3, R_4, R_5, R_6, R_7)$=(0.7257,0.927,0.81,0.8743,0.7780.762,0.8248)

我们采用 F1 值来评判模型的优劣，经过计算得到随机森林模型的 F1 值为 0.81。

5.1.3.2 支持向量机模型

沿用随机森林模型使用的训练集和测试集。通过参数寻优来构造支持向量机模型。在特征空间划出一个超平面，利用间隔最大化距离来得到最优分离超平面:

$$w^T \cdot x_i + b = 0$$

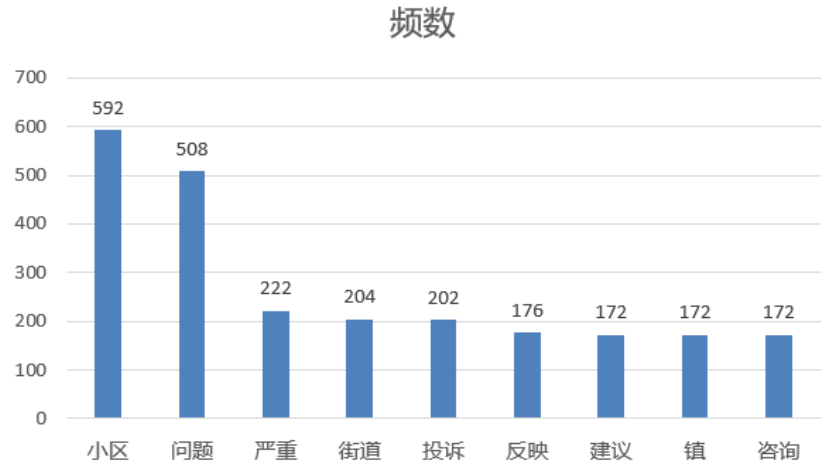
同时得到分类决策函数f(x)，其公式在模型建立中已给出。

利用 R 语言编程对模型进行求解，得到真实值和预测值的混淆矩阵（矩阵见附录）并通过混淆矩阵计算查准率和查全率，计算得到 F1 值为 0.7。和随机森林的 F1 值 0.81 进行比较，认为随机森林方法更优。

5.2 问题二模型求解

5.2.1 清洗文本

下图体现了留言主题中出现词频最高的前 10 个词。



从词频统计的结果来看，该网站的留言基本上是群众的一投诉些社区问题，并给予一些自己的建议，这与题目要求我们完成热点问题分类的要求也相呼应。

5.2.2 文本聚类

生成词频矩阵寻找文本相似度的过程与问题一基本一直。而文本聚类的过程如下：

1）首先我们计算出总的文本分得词量 M，将每篇留言看成 M 维空间的一个点。

2）在 M 维空间中随机选取 10 个点作为初始簇中心点；

3）计算每个点分别到 10 个簇中心点的欧式距离，将每个点划分至与其欧氏距离最近的质心点簇中；

4）计算新的簇内偏差，并根据每个点的坐标将所有维度相同的值相加求每个维度的平均值，得到新的簇中心点；

5）计算新簇内偏差的值，若小于 1 或进行了 100 次迭代计算则退出，否则重新迭代；

6）得到所有的中心结果。

最终，我们将所有相同主题的留言数量统计起来并进行排序，重新将附件 3 按照主题分类进行排序。

5.3 问题三求解

5.3.1 答复意见量化处理

1）相关性方面：若该答复意见是在解释该问题则相关性得分为 1；否则，为 0。

2）完整性方面：我们认为答复意见的长度与完整性密切相关，若答复意见的文字长度为 200 字一下，则完整性得分为 1；200 字到 400 字之间完整性得分为 2；400 字以上完整性得分为 3。

3）可解释性方面：若该答复中包含具体的解决方法，则可解释性得分加 1；若该答复中包含具体的相关处理部门或人员，则可解释性得分加 1；若该答复中包含具体的时间，则可解释性得分加 1；若该回复内容均不满足以上三条，则可解释性得分为 0。

通过这三条将回答的质量进行了量化处理，处理后的答复意见的质量得分可能为 1，2，3，4，5，6，7 分。即认为答复质量的等级为一到七级。

其中：

质量得分=相关性得分+完整性得分+可解释性得分

联系相关性、完整性、可解释性的三个量化标准，对附件 4 的答复文本内容的前 20 项数据进行了量化分析，得到如下结果：

序号	相关性得分	完整性得分	可解释性得分	总得分
1	1	3	2	6
2	1	2	3	6
3	1	2	1	4
4	1	2	3	6
5	1	1	2	4
6	1	1	1	3
7	1	2	3	6
8	1	3	3	7
9	1	3	2	6
10	1	2	0	3
11	1	3	3	7
12	1	2	3	6
13	1	1	3	5
14	1	1	1	3
15	1	1	3	5
16	1	3	3	7
17	1	1	0	2
18	1	1	2	4
19	1	1	3	5
20	1	1	0	2

5.3.2 随机森林模型测试

我们将答复意见进行了分类，整体的答复意见质量共有七类。利用第一问的分类思想，我们考虑将以上 20 个数据作为训练文本，然后利用随机森里模型对其他的答复意见进行测试，计算出他们的分类。

若通过模型进行测试的结果与通过相关性、完整性、可解释性判断的结果的差异过大，我们将增大训练文本的数量，不断地完善该模型，从而提高其判断答复意见质量的正确率。

六、模型检验

6.1 随机森林模型

6.1.1 灵敏度和特异度

我们使用特异性（specificity）和灵敏度（sensitivity）这两个指标来描述分类器的性能。特异度表示的是所有负例中被分对的比例，度量了分类器对负例的判断效果。

$$S_p = \frac{TN}{TN + FP}$$

灵敏度则度量分类器对正例的判断效果。

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

通过计算得到随机森林模型的灵敏度和特异度分别为 0.9 和 0.71，灵敏度和特异度其值越接近于 1，说明其诊断准确性越好。所以可以认为该模型分类效果较好。

七、模型评价

模型优缺点：

优点

1、构建的随机森林模型训练速度很快，准确度高。

2、随机森林模型对于高维数据有很好的的分类性能且不易过拟合。

3、基于 TF-IDF 方法的向量空间模型可以将文本处理成准确的向量数据，为后续的结题提供了很大的帮助。

缺点：

1、 以主题分类为主要目标，首先没有明确时间段的限制性，故热点问题的时间范围相对而言较大。

2、量化回复质量的指标得分不够具体，不能准确衡量回复内容的质量。

八、参考文献

[1]Tom M Mitchell.机器学习[M].曾华军，张银奎译.北京:机械工业出版社, 2003

[2]Breiman L.Random forests[J].Machine Learning, 2001, 45 (1) :5~32.

[3]赵苏;李秀;刘文煌.基于分类器性能评价的 Bagging 文本分类算法[J].计算机工程,2008,No.294,67-69.

[4]贺捷.随机森林在文本分类中的应用[C].华南理工大学,2015.

[5]孙源;胡志军.基于高频词和 AUC 优化的随机森林文本分类模型[J].数学的实践与认识,2020,v.50,12-17.

[6]姜博闻.基于向量空间模型的文本分类及 R 语言实现[C].山东师范大学,2018.

[7]赵鹏.基于 BC-ACO 模型的文本分类技术研究与应用[C].电子科技大学,2016.

[8]王吉俐;彭敦陆;陈章;刘丛.AM-CNN:一种基于注意力的卷积神经网络文本分类模型[J].小型微型计算机系统,2019,v.40,24-28.

[9]刘宁;陈凌云;熊文涛.基于文本挖掘的网络热点舆情分析——以问题疫苗事件为例[J].湖北工程学院学报,2019,v.39;No.183,62-66.

[10]周森;邓婕.基于文本分析的互联网金融投诉热点问题探讨——以第三方支付为例[J].中国市场,2018,No.989,197-199.

[11]孙源;胡志军.基于高频词和 AUC 优化的随机森林文本分类模型[J].数学的实践与认识,2020,v.50,12-17.

[12]李钰曼;陈志泊;许福.基于 KACC 模型的文本分类研究[J].数据分析与知识发现,2019,v.3;No.34,93-101.

九、附录

```
###预处理
setwd("C:/Users/Dell/Desktop/泰迪")
library('openxlsx')
x <- read.xlsx('附件 2.xlsx')
x1 <- x[x[,6]=='城乡建设',]
x11 <- x1[,5]
```

```
library(Rwordseg)
#installDict("自然语言处理及计算语言学相关术语.scel",'computer')
#
library(jiebaR)
library(wordcloud)
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x11 <- x11[x11!=" "]
##分词，并将分词结果转换为向量
x11 <- unlist(lapply(X = x11,FUN = segmentCN))

#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
   stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
   x1 <- subset(x1,x1!=stopwords_CN[j])
}
seg <- table(x1) #统计词频
length(seg) #查看处理完后剩余的词数
d <- sort(seg,decreasing=T)
key1 <- d[1:50]
key1
#分组
e <- unique(x[,6])
y <- list()
```

```
for (i in 1:7) {
  y[[i]] <- x[x[,6]==e[i],]
}
###分词及特征词
#####################################
#####################################
#####################################
x12 <- y[[2]][,5]
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x12 <- x12[x12!=" "]
##分词，并将分词结果转换为向量
x12 <- unlist(lapply(X = x12,FUN = segmentCN))
#预处理
a <- x12[nchar(x12)>1] #去除字符长度小于 2 的词语
x2 <- gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",a)
#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
  stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
  x2 <- subset(x2,x2!=stopwords_CN[j])
}
seg2 <- table(x2) #统计词频
length(seg2) #查看处理完后剩余的词数
d2 <- sort(seg2,decreasing=T)
d2
```

```
key2 <- d2[1:50]
key2
#####################################
#####################################
#####################################
```

环境保护

```
x13 <- y[[3]][,5]
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x13 <- x13[x13!=" "]
##分词，并将分词结果转换为向量
x13 <- unlist(lapply(X = x13,FUN = segmentCN))
#预处理
a <- x13[nchar(x13)>1] #去除字符长度小于 2 的词语
x3 <- gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",a)
#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
  stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
  x3 <- subset(x3,x3!=stopwords_CN[j])
}
seg3 <- table(x3) #统计词频
length(seg3) #查看处理完后剩余的词数
d3 <- sort(seg3,decreasing=T)
key3 <- d3[1:50]
key3
```

```
#####################################
#####################################
#####################################
```

交通运输

```
x13 <- y[[3]][,5]
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x13 <- x13[x13!=" "]
##分词，并将分词结果转换为向量
x13 <- unlist(lapply(X = x13,FUN = segmentCN))
#预处理
a <- x13[nchar(x13)>1] #去除字符长度小于 2 的词语
x3 <- gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",a)
#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
  stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
  x3 <- subset(x3,x3!=stopwords_CN[j])
}
seg3 <- table(x3) #统计词频
length(seg3) #查看处理完后剩余的词数
d3 <- sort(seg3,decreasing=T)
key3 <- d3[1:50]
key3
```

```
#####################################
#####################################
#####################################
```

教育文体

```
x14 <- y[[4]][,5]
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x14 <- x14[x14!=" "]
##分词，并将分词结果转换为向量
x14 <- unlist(lapply(X = x14,FUN = segmentCN))
#预处理
a <- x14[nchar(x14)>1] #去除字符长度小于 2 的词语
x4 <- gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",a)
#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
   stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
   x4 <- subset(x4,x4!=stopwords_CN[j])
}
seg4 <- table(x4) #统计词频
length(seg4) #查看处理完后剩余的词数
d4 <- sort(seg4,decreasing=T)
key4 <- d4[1:50]

#####################################
#####################################
```

```
#####################################
```

劳动和社会保障

```
x15 <- y[[5]][,5]
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x15 <- x15[x15!=" "]
##分词，并将分词结果转换为向量
x15 <- unlist(lapply(X = x15,FUN = segmentCN))
#预处理
a <- x15[nchar(x15)>1] #去除字符长度小于 2 的词语
x5 <- gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",a)
#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
  stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
  x5 <- subset(x5,x5!=stopwords_CN[j])
}
seg5 <- table(x5) #统计词频
length(seg5) #查看处理完后剩余的词数
d5 <- sort(seg5,decreasing=T)
key5 <- d5[1:50]
key5

#####################################
#####################################
#####################################
```

商贸旅游

```
x16 <- y[[6]][,5]
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x16 <- x16[x16!=" "]
##分词，并将分词结果转换为向量
x16 <- unlist(lapply(X = x16,FUN = segmentCN))
#预处理
a <- x16[nchar(x16)>1] #去除字符长度小于 2 的词语
x6 <- gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",a)
#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
  stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
  x6 <- subset(x6,x6!=stopwords_CN[j])
}
seg6 <- table(x6) #统计词频
length(seg6) #查看处理完后剩余的词数
d6 <- sort(seg6,decreasing=T)
key6 <- d6[1:50]
key6
d6

#######################################
#######################################
#######################################
```

卫生计生

```
x17 <- y[[7]][,5]
#预处理，这步可以将读入的文本转换为可以分词的字符，没有这步不能分词
x17 <- x17[x17!=" "]
##分词，并将分词结果转换为向量
x17 <- unlist(lapply(X = x17,FUN = segmentCN))
#预处理
a <- x17[nchar(x17)>1] #去除字符长度小于 2 的词语
x7 <- gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",a)
#停止词
library(readr)
library(dplyr)
data_stw <- read.xlsx('停止词.xlsx')
stopwords_CN=c(NULL)
for(i in 1:dim(data_stw)[1]){
   stopwords_CN=c(stopwords_CN,data_stw[i,1])
}
for(j in 1:length(stopwords_CN)){
   x7 <- subset(x7,x7!=stopwords_CN[j])
}
seg7 <- table(x7) #统计词频
length(seg7) #查看处理完后剩余的词数
d7 <- sort(seg7,decreasing=T)
key7 <- d7[1:50]
key7[1]
cl <- rbind(key1,key2,key3,key4,key5,key6,key7)
write.xlsx(cl,"C:/Users/Dell/Desktop/泰迪/特征词 350 个.xlsx")

####################
####################
```

随机森林

```
write.xlsx(e,'C:/Users/Dell/Desktop/泰迪/e.xlsx')
x <- read.xlsx('e.xlsx')
row <- nrow(x)
m1 <- sample(row,0.7*row)
train<- x[m1,]
test <- x[-m1,]
library("randomForest")
train$一级标签  <- as.factor(train[,243])
test$一级标签<-as.factor(test[,243])
rFM<-randomForest(一级标签
~.,data=train,importance=TRUE,proximity=TRUE);
pre <- predict(rFM,test)
#真实值和预测值整合在一起
comb <- data.frame(prob=pre,obs=test$一级标签);comb
#输出混淆矩阵
table(test$一级标签,pre,dnn = c("真实值","预测值"))
library('pROC')
ran <- roc(test$一级标签,as.numeric(pre))
plot(ran,print.auc=T,auc.polygon=T,grid=c(0.1,0.2),grid.col=c("green","red"),max.
auc.polygon=T,auc.polygon.col="skyblue",main='随机森林算法 ROC 曲线')

####################
####################
```

支持向量机

```
svm<- svm(一级标签  ~ .,
                data = train,
                type = 'C',kernel = 'radial' )
svm
pre_svm <- predict(svm,newdata = test)
```

```
obs_p_svm = data.frame(prob=pre_svm,obs=test$一级标签)
obs_p_svm
comb <- data.frame(prob=pre_svm,obs=test$一级标签);comb
#输出混淆矩阵
w <- table(test$一级标签,pre_svm,dnn = c("真实值","预测值"))
```

#svm 的混淆矩阵

```
> w
```

	预测值						
真实值	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
城乡建设	336	4	18	16	26	12	7
环境保护	32	193	3	10	5	2	3
交通运输	10	2	133	1	9	8	0
教育文体	10	1	1	327	27	17	2
劳动和社会保障	24	3	4	11	397	4	25
商贸旅游	45	3	5	3	19	151	1
卫生计生	6	2	0	6	27	4	179

#词频矩阵

TF-IDF codes

```
library('jiebaR')
library('openxlsx')
x <- as.matrix(read.xlsx('附件 2.xlsx'))
uni <-read.xlsx('C:\\Users\\xunchun\\Desktop\\特征词-类别.xlsx')
install.packages('tm')
install.packages('tmcn')
install.packages('Rwordseg')
install.packages('wordcloud')
install.packages('dplyr')
install.packages('stringr')
install.packages('tidytext')
install.packages('tidyr')
install.packages('proxy')
library(tm)
library(tmcn)
library(Rwordseg)
library(wordcloud)
```

```
library(dplyr)
library(stringr)
library(tidytext)
library(tidyr)
library(proxy)
#str(x)
#文本清洗
for(i in 1:nrow(x)){
   x[i,3] <- gsub('[a-zA-Z]',",x[i,3])
   x[i,3] <- gsub("\t","",x[i,3])
   x[i,3] <- gsub("？ ","",x[i,3])
   x[i,3] <- gsub("?","",x[i,3])
   x[i,3] <- gsub(",","",x[i,3])
   x[i,3] <- gsub(",","",x[i,3])
   x[i,3] <- gsub("[0-9]","",x[i,3])
}

for(i in 1:nrow(x)){
   x[i,5] <- gsub('[a-zA-Z]',",x[i,5])
   x[i,5] <- gsub("\t","",x[i,5])
   x[i,5] <- gsub("？ ","",x[i,5])
   x[i,5] <- gsub("?","",x[i,5])
   x[i,5] <- gsub(",","",x[i,5])
   x[i,5] <- gsub(",","",x[i,5])
   x[i,5] <- gsub("[0-9]","",x[i,5])
}
theme <- tibble(line=1:nrow(x),text=x[,3])
theme <- unnest_tokens(theme,word,text)   #保留行号，删除标点，大写改小写
detail <- tibble(line=1:nrow(x),text=x[,5])
detail <- unnest_tokens(detail,word,text)
```

```
#删除停用词
data("stop_words")
theme <- theme%>%
   anti_join(stop_words)
detail <- detail%>%
   anti_join(stop_words)

theme%>%
   count(word,sort=T)#统计词出现频数
detail%>%
   count(word,sort=T)#统计词出现频数
word_x <- uni[,-3] #去掉频数
#筛选主题一,filter 也是筛选

renum <- theme%>%
   count(line)#计算句子的长度

result1 <- theme%>%
   inner_join(word_x)
result2 <- detail%>%
   inner_join(word_x)

b <- table(result1$line,result1$word);b #二维列联表
b <- as.matrix(b);b

renum <- as.matrix(renum);renum
tf <-
b[which(rownames(b)%in%renum[,1]),]/(renum[which(rownames(b)%in%renum[,1])
,2]);b
```

```
tf <- t(tf);tf
idf <- log(ncol(tf)/(1+rowSums(tf!=0)));idf
idf <- diag(idf);idf
tf_idf <- crossprod(tf,idf)
colnames(tf_idf) <- rownames(tf)
tf_idf
matrix_theme <- tf_idf / sqrt( rowSums( tf_idf^2 ) )
write.xlsx(matrix_theme,"矩阵-theme.xlsx",row.names=T)

matrix_detail <- tf_idf / sqrt( rowSums( tf_idf^2 ) )
write.xlsx(matrix_detail,"矩阵-detail.xlsx",row.names=T)
```