互联网文本挖掘的综合评价

摘要

随着网络问政平台的普及，政府收集到的民意的文本数据不断攀升。从各类社情民意相关的文本数据中挖掘出有用信息，对互联网的留言以及群众的反馈进行实时、准确、高效的评价尤其重要。同时为了方便相关部门的工作，利用大数据、云计算、人工智能等技术，建立基于**自然语言**处理技术的智慧政务系统，对提升政府的管理水平和施政效率具有极大的推动作用。

针对问题 1，首先采用卷积神经网络对文本进行分类。首先用 **jieba** 进行文本预处理，即将文本数据分割称若干词语，并去掉一些无意义的词语和符号。其次使用 **Tokenizer** 对词语进行**独热编码**和词嵌入，将其向量化，转换成数字信息，将每条文本转变成一个向量。然后使用改进的**卷积神经网络**，构建 **keras 和 Textcnn 模型**对训练集和测试集进行模型训练，同时为了使其在测试集上有较好的泛化性，将该模型进行了简化，其中测试集的准确率高达 0.99。最后计算 F1=0.84，得到该分类方法较好的结论。

针对问题 2，首先对附件 3 中的留言信息按照特定的地点进行筛选出“区”“县”以及除去 A 市的其他市，对剩下的 A 市和不明确的地点使用 **K-Means 算法**，从留言文本中挖掘出地点关键词，对它们进行**聚类分析**，从“A 市”“区”“县”三个方面进行分类，其中大部分数据集中在“A 市”，考虑到在问题一中一级标签有 7 类，故将其分为 7 类。最后得到热点排名前 5 的问题，问题 ID 分别为 10、2、1、22、3，其中问题 ID 为 10 的热度指数高达 721。

针对问题 3，首先对附件 4 中的数据依次进行相关性、完整性、可解释性评价，用 **TF-IDF 算法**计算文本相似度来反映，得到平均的评价系数为 0.155，可以看出政府的多数回复较为让群众满意，但仍存在一些没有提供任何帮助的回复。

对“智慧文本”数据挖掘评价模型研究，为了对留言文本进行系统规范的分析以及评价，本文采用了多种方法，并且具有良好的扩展性，构建文本挖掘模型以此达到准确、高效的处理文本数据的作用，并给予充分、良好的评价，从而提升政府与群众的亲切度和提高政府的管理水平。

关键词：独热编码，卷积神经网络，K-Means 算法，聚类分析，TF-IDF 算法

一、挖掘目标

1.1 挖掘背景

近年来，随着微信、微博、市长信箱、阳光热线等网络问政平台逐步成为政府了解民意、法汇聚民智、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。

1.2 挖掘目标

问题 1：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（参考附件 1 提供的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。请根据附件 2 给出的据剧，建立关于留言内容的一级标签分类模型。

问题 2：某一时段内群众集中反映的某一问题可称为热点问题，如“XXX 小区多位业主多次反映:入夏以来小区楼下烧烤店深夜经营导致噪音和油烟扰民”。及时发现热点问题，有助于相关部门进行有针对性地处理，提升服务效率。请根据附件 3 将某一时段内反映特定地点或特定人群问题的留言进行归类，定义合理的热度评价指标，并给出评价结果，按表 1 的格式给出排名前 5 的热点问题，并保存为文件“热点问题表.xls”。按表 2 的格式给出相应热点问题对应的留言信息，并保存为“热点问题留言明细表.xls”。

问题 3：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 问题一分析

本文考虑采用独热编码和字嵌入将附件 2 中的留言文本信息转换成数字信息。具体实现方法是首先利用 NLP 将自然语言组成的训练文本表示成离散的数据格式，即将每一句留言信息拆解为若干小段文字，然后计算每一小段文字出现的概率。构建 textCNN 模型来对数据进行训练与测试，找到最合适的处理方式。最后利用 F-score 方法对分类方法评价。

2.2 问题二分析

本文首先提取出留言信息本题根据留言的地点信息进行筛选，从附件 3 中的留言中挖掘出地点信息，发现有关“A 市”的留言信息较多，于是首先找出县级和区级的地点进行分类，再单独对 A 市地点进行分类，考虑到在问题一中一级标签有七类，故将其分为 7 类。

2.3 问题三分析

本文从三个方面依次进行评价分析，考虑到 TF-IDF 算法可以计算文本的相似度，利用相似度来评价政府对群众问题的回复。

三、符号说明

符号	含义
P_i	第 i 类的查准率
R_i	第 i 类的查全率

四、模型假设

假设一：所给数据均是在一定时间内全部发生的事情；

假设二：不考虑所给数据中没有的信息而导致对评价方案产生影响的因素；

假设三：政府所给的回复意见中不考虑情感因素的影响。

五、数据预处理与问题求解

5.1 问题一　群众留言分类

5.1.1　名词解释

(1) 卷积神经网络（CNN）

卷积神经网络（Convolutional Neural Network, CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。

Yoon Kim 在 2014 年“Convolutional Neural Networks for Sentence Classification”论文中提出 TextCNN（利用卷积神经网络对文本进行分类的算法），如图

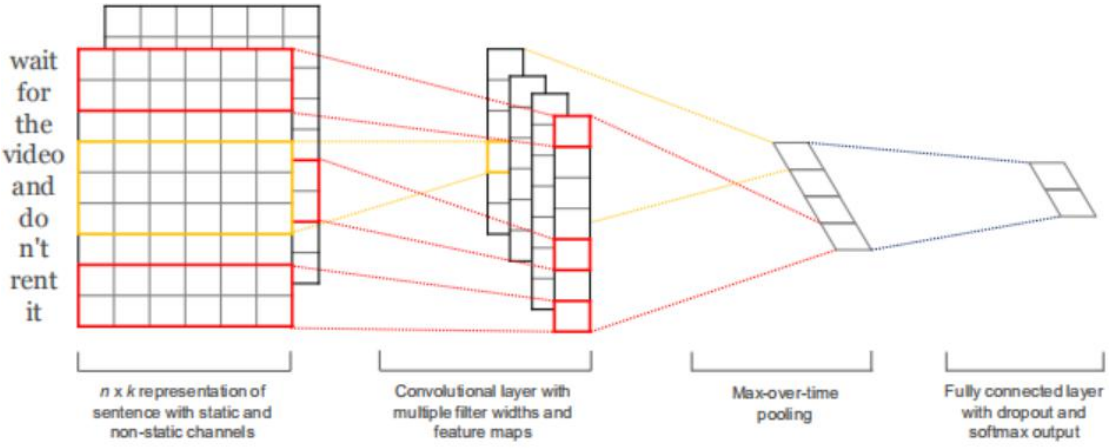

图 1 双通道模型架构

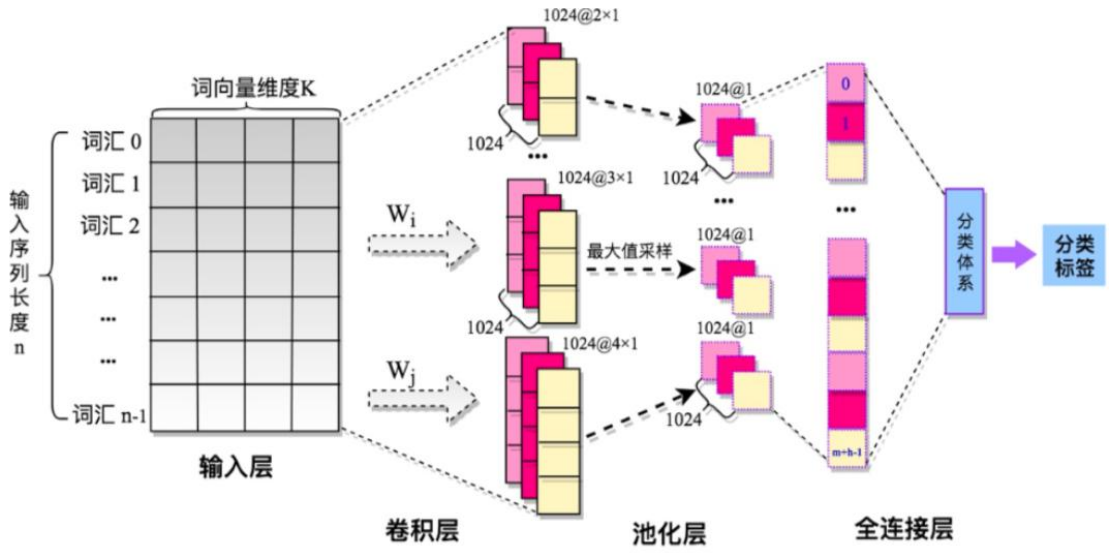

图 2 Text-CNN 过程

上图很好地诠释了模型的框架。假设我们有一些句子需要对其进行分类。句子中每个词是由 n 维词向量组成的，也就是说输入矩阵大小为 m*n，其中 m 为句子长度。CNN 需要对输入样本进行卷积操作，对于文本数据，filter 不再横向滑动，仅仅是向下移动，有点类似于 N-gram 在提取词与词间的局部相关性。图中共有三种步长策略，分别是 2,3,4，每个步长都有两个 filter（实际训练时 filter 数量会很多）。在不同词窗上应用不同 filter，最终得到 6 个卷积后的向量。然后对每一个向量进行最大化池化操作并拼接各个池化值，最终得到这个句子的特征表示，将这个句子向量丢给分类器进行分类，至此完成整个流程。

(2) 独热编码（One-Hot）

独热编吗又称为一位有效编码，主要是采用 N 位状态寄存器来对 N 个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效。

One-Hot 编码是分类变量作为二进制向量的表示。这首先要求将分类值映射到整数值。然后，每个整数值被表示为二进制向量，除了整数的索引之外，它都是零值，它被标记为 1。例如对六个状态进行编码：

自然顺序码为 0,1,2,3,4,5

独热编码则是 000001,000010,000100,001000,010000,100000

可以这样理解，对于每一个特征，如果它有 m 个可能值，那么经过独热编码后，就变成了 m 个二元特征（如成绩这个特征有好，中，差变成 one-hot 就是（100, 010, 001）。并且，这些特征互斥，每次只有一个激活。因此，数据会变成稀疏的。这样做的好处主要有：一是解决了分类器不好处理属性数据的问题；

二是在一定程度上也起到了扩充特征的作用。

(3) 词嵌入

词嵌入将字表示为密集字向量（也称为字嵌入），其训练方式不像独热编码那样，这意味着词嵌入将更多的信息收集到更少的维度中。

嵌入词并不像人类那样理解文本，而是映射语料库中使用的语言的统计结构，其目标是将语义意义映射到几何空间，该几何空间也被称为嵌入空间（embedding space）。这个研究领域的一个著名例子是能够映射 King - Man + Woman = Queen。

怎么能获得这样的词嵌入呢？这里有两种方法，其中一种是在训练神经网络时训练词嵌入层。另一种方法是使用预训练好的词嵌入。

现在，需要将数据标记为可以由词嵌入使用的格式。Keras 为文本预处理和序列预处理提供了几种便捷方法，我们可以使用这些方法来处理文本。

首先，可以从使用 Tokenizer 类开始，该类可以将文本语料库向量化为整数列表。每个整数映射到字典中的一个值，该字典对整个语料库进行编码，字典中的键是词汇表本身。此外，可以添加参数 num_words，该参数负责设置词汇表的大小。num_words 保留最常见的单词。

(4) NLP 概率

NLP (Natural Language Processing) 是人工智能的自然语言处理。它是人工智能的一个子领域。自然语言是人类智慧的结晶，自然语言处理是人工智能中最为困难的问题之一，而对自然语言处理的研究也是充满魅力和挑战的。NLP 概率是用来处理自然语言的，它关心某个句子出现的概率。

5.1.2 TextCNN 文本分类

TextCNN 是利用卷积神经网络对文本进行分类的算法。深度学习模型在计算机视觉与语音识别方面取得了卓越的成就，在 NLP 领域也是可以的。将卷积神经网络 CNN 应用到文本分类任务，利用多个不同 size 的 kernel 来提取句子中的关键信息，从而能够更好地捕捉局部相关性。

文本分类是自然语言处理领域最活跃的研究方向之一，目前文本分类在工业界的应用场景非常普遍，从新闻的分类、商品评论信息的情感分类到微博信息打标签辅助推荐系统，了解文本分类技术是 NLP 初学者比较好的切入点，较简单

且应用场景高频。

文本预处理是为了将每个样本转换为一个数字矩阵，矩阵的每一行表示一个词向量。下面用 Keras 进行文本预处理：

Step1：对数据集进行处理使得词与词之间以空格分开

对文本数据进行分割成若干词语。中文文本分类需要分词，有很多开源的中文分词工具，例如 Jieba 等。分词即对文本做进一步的处理，去除掉一些高频词汇和低频词汇，去掉一些无意义的符号等。然后是建立词典以及单词索引，建立词典就是统计文本中出现多少了单词，然后为每个单词编码一个唯一的索引号，便于查找。以下是分词后的部分结果：

```
9198    K3 县 下属 卫生室 医疗机构 执业 许可证 过期 三年 换证 去年 19 年 网友 向市...
9199    收敛 蒙混过关 药品 招标 依旧 涉嫌 黑幕 西地省 抗菌 药物 带量 采购 2019 年 ...
9200     艾滋病 血液 传播 传播方式 日常 接触 拥抱 接吻 传播 血液 尿液 唾液 艾滋病 检测 准确
9201              我开 小型 餐馆 食品药品 监督局 办个 卫生 许可证 3500       G5 县
9202          尊敬 领导 您好 一名 妇科病 患者 2014 年 24 号 网上 预约 中南大学 ...
9203    张 厅长        基层 卫生 工作 提些 建议        抓 落实 现有 政策方针 工作 制度...
9204          强烈建议 医院 手术室 装上 高清 摄像机 远程 传输 卫生 行政 管理 部门 图像...
9205          夫妻 农村户口 女 岁 岁 15 斤 治疗 两年 一级 脑瘫 女户 招郎 男方 两 ...
9206          2015 年 16 号 B 市中心 医院 做 无痛 人流 手术 手术 怀孕 症状 2...
9207                              再婚 想 小孩 不知 我省 二胎 新 政策 先 怀孕 做
9208    K8 县惊现 奇葩 证明        西地省 K8 县人 想 生二孩 告知 要开 证明 没生 二...
9209          领导 你好 未 婚生子 2013 年 接受 处罚 小孩 上户 小孩 外地 上学 需 ...
```

图 3 分词后的部分结果

Step2：利用 Tokenizer 对词进行编码

使用 Keras 的 Tokenizer 模块实现转换。当我们创建了一个 Tokenizer 对象后，使用该对象的 fit_on_texts()函数，可以将输入的文本中的每个词编号，编号是根据词频的，词频越大，编号越小。使用 word_index 属性可以看到每次词对应的编码。

Step3：将文本数据转换成数字特征

将数据集中的每条文本转换为数字列表，使用每个词的编号进行编号。使用该对象的 texts_to_sequences（）函数，将每条文本转变成一个向量。

Step4：截长补短是的所有样本长度一致

使用 pad_sequences()让每句数字影评长度相同。由于每句话的长度不唯一，需要将每句话的长度设置一个固定值。将超过固定值的部分截掉，不足的在最前面用 0 填充。

Step5：对一级标签进行独热编码

一级分类一共有 7 个分类，它们的编码如下表

表 1 一级标签编码情况

城乡建设	1000000
环境保护	0100000
交通运输	0010000
教育文体	0001000
劳动和社会保障	0000100
商贸旅游	0000010
卫生计生	0000001

Step6：Keras 文本预处理代码实现

Step7：使用 Embedding 层将每个词编码转换为词向量

Embedding 层基于上文所得的词编码，对每个词进行 one-hot 编码，每个词都会是一个 vocabulary_size 维的向量；然后通过神经网络的训练迭代更新得到一个合适的权重矩阵，行大小为 vocabulary_size，列大小为词向量的维度，将本来以 one-hot 编码的词向量映射到低维空间，得到低维词向量。需要声明一点的是 Embedding 层是作为模型的第一层，在训练模型的同时，得到该语料库的词向量。当然，也可以使用已经预训练好的词向量表示现有语料库中的词。

Step8：模型训练

Step9：模型预测

5.1.3 基于 keras 和 Textcnn 模型的构建、训练与测试

基础版 CNN（模仿 LeNet-5）

LeNet-5 是卷积神经网络的作者 Yann LeCun 用于 MNIST 识别任务提出的模型。模型很简单，就是卷积池化层的堆叠，最后加上几层全连接层。将其运用在文本分类任务中。

然而该模型过于复杂。它在训练集上表现很好，但是泛化能力比较差，在测试集上表现不好，如图：

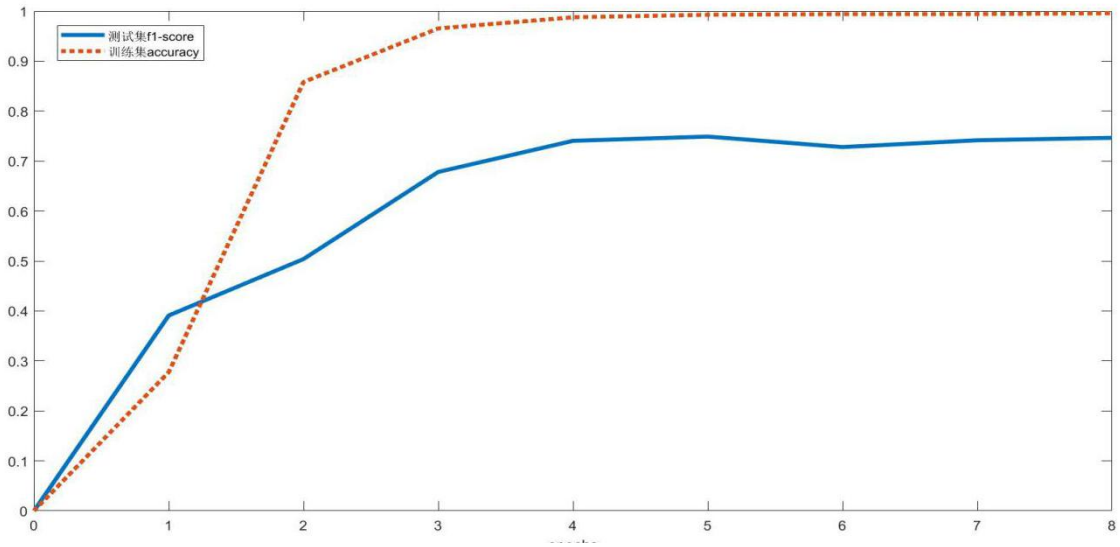

图 4 过拟合图

其中每个红点为训练集的训练实例，红色曲线是我们训练得出的方程，可以看出红色曲线最终无限接近 1，即训练的模型能够完美的拟合训练集的实例，但是蓝色的曲线最终无限接近 0.7，在测试集表现比较差，泛化能力很差。

于是对传统模型 LeNet-5 做了简化，用简单版 TextCNN 处理数据，发现在测试集表现效果较好。

5.1.4 评价系数

通常使用 F-Score 对分类方法进行评价：

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

查准率(precision)与查全率(recall)的解释：对于二分类问题，可以将样例根据其真实类别与分类器预测类别的组合划分为以下四种情形：

- 真正例(true positive)：预测为正，真实也为正
- 假正例(false positive)：预测为正，但真实为反
- 真反例(true negative)：预测为反，真实也为反
- 假反例(true negative)：预测为反，但真实为正

我们令 TP、FP、TN、FN 分别表示其对应的样例数，分类结果的 “混淆矩阵”如下表所示：

表 2 分类结果

真实情况	预测结果	
	正例	反例
正例	TP（真正例）	FN（假反例）
反例	FP（假正例）	TN（真反例）

显然，TP+FP+TN+FN = 样例总数。从而查准率 P、查全率 R 分别定义为：

- $P = \frac{TP}{TP + FP}$ 也就是：所有预测为正的样例中，真的正例所占比例
- $R = \frac{TP}{TP + FN}$ 也就是：所有真实为正的样例中，真的正例所占比例

查准率与查全率是一对矛盾的度量。一般来说，查准率高时，查全率往往偏低；查全率高时，查准率往往偏低。通常只有在一些简单任务中，才能使查准率和查全率都很高。

F1 是基于查准率与查全率的调和平均定义的：

$$\frac{1}{F_1} = \frac{1}{2} * (\frac{1}{P} + \frac{1}{R})$$

调和平均数又称倒数平均数，是总体各统计变量倒数的算术平均数的倒数。于是有 F1 的计算公式：

$$F_1 = \frac{2PR}{P + R}$$

这是二分类问题计算 F1 的公式，将其推广到多分类问题，就得到对该题分类方法的评价公式：

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

最后计算结果为 0.844394，表明该分类方法效果较好。

5.2 问题二 热点问题挖掘

5.2.1 基于 K-Means 算法的文本聚类

K-Means 算法

K-Means 算法是输入聚类个数 k，以及包含 n 个数据对象的数据库，输出满足方差最小标准 k 个聚类的一种算法。k-means 算法接受输入量 k ；然后将 n 个数据对象划分为 k 个聚类以便使得所获得的聚类满足:同一聚类中的对象相似度较高;而不同聚类中的对象相似度较小。聚类相似度是利用各聚类中对象的均值所获得一个“中心对象”(引力中心)来进行计算的。

K-Means 算法工作过程是：首先从 n 个数据对象任意选择 k 个对象作为初始聚类中心；而对于所剩下其它对象，则根据它们与这些聚类中心的相似度(距离)，分别将它们分配给与其最相似的(聚类中心所代表的)聚类；然后再计算每个所获新聚类的聚类中心(该聚类中所有对象的均值)；不断重复这一过程直到标准测度函数开始收敛为止。一般都采用均方差作为标准测度函数. k 个聚类具有以下特点:各聚类本身尽可能的紧凑，而各聚类之间尽可能的分开。

本题将基于 K-Means 算法的文本聚类分析，从网上群众留言信息中挖掘出地点关键词，并将它们进行聚类。根据附件 3 的数据，先找出区级和县级的地点信息，剩余的大部分均是有关“A 市”的信息，按照这三级进行分类，操作流程如下：

（1）读取给定数据集内容；

（2）对文本内容进行预处理，包括去标点和使用结巴分词；

（3）用 Scikit-learn 对文本内容进行 tfidf 计算并构造 N*M 矩阵（N 个文档，M 个特殊词）；

（4）使用 K=means 进行文本聚类（省略特征词过来降维过程）；

（5）对测试集进行分类。

概要设计如下：

step1 读取训练集文本内容

读取给定数据集文件夹中每一个文档后，将文本内容写入一个 Result.txt，每一行为一个文档，方便后面词频矩阵的处理。

step2 文本预处理

读取之前存放所有文本内容的 Result.txt，将其内容去空格，去标点，并用结巴进行分词。

step3　筛选

按照“区”“县”“A 市”三个方面将留言文本信息筛选出来。

step4　特征提取

使用 scikit-learn 工具调用 CountVectorizer()和 TfidfTransformer()函数计算 TF-IDF 值，将文本转为词频矩阵，矩阵元素 a[i][j] 表示 j 词在 i 类文本下的词频。将词频矩阵保存在 TF-IDF_Result 文档中。

step5　K-Means 聚类

对文本根据浏览内容后的经验分为 3 类，调用 sklearn.cluster 实现，并保存该聚类模型，对测试集使用。

step6　测试集分类

使用 clf.fit_predict 方法测试测试集文本。

5.2.2　优点分析

K-Means 优点有：

（1） 原理比较简单，实现也是很容易，收敛速度快。

（2） 当结果簇是密集的，而簇与簇之间区别明显时，它的效果较好。

（3） 主要需要调参的参数仅仅是簇数 k。

5.3 问题三　答复意见的评价

本题利用 TF-IDF 算法对综合答复的相关性、完整性以及可解释性进行综合评价。

5.3.1　TF-IDF 算法

TF 是指词频，即在文章中出现次数最多的词，但是在很多情况下出现次数较多的词并不一定就是关键词，比如常见的对文章本身并没有多大意义的停用词。所以我们需要一个重要性调整系数来衡量一个词是不是常见词。该权重为 IDF（逆文档频率），它的大小与一个词的常见程度成反比。在我们得到词频（TF）和逆文档频率（IDF）以后，将两个值相乘，即可得到一个词的 TF-IDF 值，某

个词对文章的重要性越高，其 TF-IDF 值就越大，TF-IDF 值较大的词就是该文本的关键词。

本题将采用计算 TF-IDF 系数的方法，来处理留言与回复的相关性，该算法步骤如下：

step1　计算词频

词频原指某个词在文章中出现的次数，但考虑到有长短之分，为了便于不同文章的比较，进行"词频"标准化。即词频为某个词在文章中出现的次数与文章的总词数之比，或某个词在文章中出现的次数与该文出现次数最多的词出现的次数之比。

step2　计算逆文档频率

利用语料库，来模拟语言的使用环境，逆文档频率计算公式为：

$$\text{IDF}=\log\left(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数}+1}\right)$$

如果一个词越常见，那么分母就越大，逆文档频率就越小越接近 0。分母之所以要加 1，是为了避免分母为 0（即所有文档都不包含该词）。log 表示对得到的值取对数。

step3　计算 TF-IDF

TF-IDF 的大小等于词频（TF）与逆文档频率（IDF）的乘积，可以看到，TF-IDF 与一个词在文档中的出现次数成正比，与该词在整个语言中的出现次数成反比。所以，自动提取关键词的算法就很清楚了，就是计算出文档的每个词的 TF-IDF 值，然后按降序排列，取排在最前面的几个词就是关键词。

TF-IDF 算法的优点是简单快速，结果比较符合实际情况，但是单纯以“词频”衡量一个词的重要性，不够全面，有时候重要的词可能出现的次数并不多，而且这种算法无法体现词的位置信息，出现位置靠前的词和出现位置靠后的词，都被视为同样重要，是不合理的。

得到分词向量后计算它们的夹角余弦值，得到平均的评价系数为 0.155，以看出政府的多数回复较为让群众满意，但仍存在一些没有提供任何帮助的回复。

六、总结

本文主要目的是利用数据挖掘和数学建模技术建立文本分析的评价模型，利用 Kearas 进行文本预处理，再对词语进行独热编码和词嵌入，用 TF-IDF 赋予权重，构建 keras 和 Texrcnn 模型对训练集和测试机进行模型训练，并不断改进模型，得到更高的准确率。另外还用了 K-Means 算法对文本进行聚类分析，得到热点问题的排名信息。最后用 TF-IDF 算法对政府的回复做出评价，提高政府的管理水平。

七、参考文献

[1] Yoon Kim.Convolutional Neural Networks for Sentence Classification[J].,2014

[2] 张彬城,陈杰彬,林越.一种基于潜在语义索引和卷积神经网络的智能阅读模型[J].,2019-02-20

[3] Asia-Lee.TextCNN 文本分类（keras 实现）[EB/OL].https://blog.csdn.net/asialee_bird/article/details/88813385,2019-03-26.

[4] Athraxon.基于 K-Means 算法的文本聚类[EB/OL].https://blog.csdn.net/EleanorWiser/article/details/70290855,2017-04-20.

[5] evan_qb.TF-IDF 权重策略[EB/OL].https://blog.csdn.net/evan_qb/article/details/78131864,2017-09-29.

[6] gg-123.大规模文本分类网络 TextCNN 介绍[EB/OL].https://blog.csdn.net/u012762419/article/details/79561441,2018-03-15.

[7] 进击的 AI 小白.【分类问题中模型的性能度量(一)】错误率、精度、查准率、查全率、F1 详细讲解[EB/OL].https://blog.csdn.net/weixin_41059350/article/details/89819844,2019-05-05.

[8] 阮一峰.TF-IDF 与余弦相似性的应用（一）：自动提取关键词[EB/OL].http://www.ruanyifeng.com/blog/2013/03/tf-idf.html,2013-03-15.

附录

问题一代码

```
import pandas as pd
import jieba
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential,Model
from keras.layers import Conv1D,MaxPooling1D,Flatten,Dropout,Embedding,BatchNormalization,Dense,concatenate
import sklearn
import keras
import numpy as np
from keras.models import load_model


sExcelFile="附件 2.xlsx"
df = pd.read_excel(sExcelFile,sheet_name='Sheet1')                    #读取 excel 文件
dflabel=df['一级标签']                                                #取出标签列
labelonehot=pd.get_dummies(dflabel)                                   #独热编码
labelonehot.shape
liuyan=df['留言主题']+df['留言详情'] #读取留言
labelshape1,labelshape2=labelonehot.shape
labelnum=(np.array(np.mat(labelonehot.values)*np.mat(np.arange(1,(labelshape2+1),1).reshape(labelshape2,1))))-1 #独热编码转换为数值
```

```
labelnum

jieba.load_userdict('newdic1.txt')
data_cut = liuyan.apply(lambda x: jieba.lcut(x))                    #分词
data_cut
stopWords = pd.read_csv('stopword.txt', encoding='GB18030', sep='hahaha', header=None)
stopWords = ['≮', '≯', '≠', '≮', ' ', '"', '月', '日', ' - ','\n','\t'] + list(stopWords.iloc[:, 0])
data_after_stop = data_cut.apply(lambda x: [i for i in x if i not in stopWords])
#去停用词
data_after_stop
adata = data_after_stop.apply(lambda x: ' '.join(x))
adata

adata.to_csv('./去停用词后.csv')

tokenizer = Tokenizer()  # 创建一个 Tokenizer 对象
# fit_on_texts 函数可以将输入的文本中的每个词编号，编号是根据词频的，词频越大，编号越小
tokenizer.fit_on_texts(adata)
vocab = tokenizer.word_index  # 得到每个词的编号

file= open('词典.txt', 'w',encoding='utf-8')
for fp in vocab:
    file.write(str(fp))
    file.write('\n')
file.close()

x_train, x_test, y_train, y_test = train_test_split(adata,labelnum, test_size=0.1)
# 将每个样本中的每个词转换为数字列表，使用每个词的编号进行编号
```

```
x_train_word_ids=tokenizer.texts_to_sequences(x_train)
x_test_word_ids =tokenizer.texts_to_sequences(x_test)

#序列模式
 # 每条样本长度不唯一，将每条样本的长度设置一个固定值
x_train_padded_seqs=pad_sequences(x_train_word_ids,maxlen=100) #将超过固定值的部分截掉，不足的在最前面用 0 填充
x_test_padded_seqs=pad_sequences(x_test_word_ids, maxlen=100)


main_input = keras.Input(shape=(100,), dtype='float64')
# 词嵌入（使用预训练的词向量）
embedder = Embedding(len(vocab) + 1, 300, input_length=100, trainable=False)
embed = embedder(main_input)
# 词窗大小分别为 3,4,5
cnn1 = Conv1D(256, 3, padding='same', strides=1, activation='relu')(embed)
cnn1 = MaxPooling1D(pool_size=48)(cnn1)
cnn2 = Conv1D(256, 4, padding='same', strides=1, activation='relu')(embed)
cnn2 = MaxPooling1D(pool_size=47)(cnn2)
cnn3 = Conv1D(256, 5, padding='same', strides=1, activation='relu')(embed)
cnn3 = MaxPooling1D(pool_size=46)(cnn3)

# 合并三个模型的输出向量
cnn = concatenate([cnn1, cnn2, cnn3], axis=-1)
flat = Flatten()(cnn)
drop = Dropout(0.2)(flat)
main_output = Dense(labelshape2, activation='softmax')(drop)
model = Model(inputs=main_input, outputs=main_output)
model.compile(loss='categorical_crossentropy',                optimizer='adam',
```

```
metrics=['accuracy'])

one_hot_labels = keras.utils.to_categorical(y_train, num_classes=labelshape2)  # 将标签转换为 one-hot 编码
model.fit(x_train_padded_seqs, one_hot_labels, batch_size=800, epochs=1)
# y_test_onehot = keras.utils.to_categorical(y_test, num_classes=3)  # 将标签转换为 one-hot 编码

result = model.predict(x_test_padded_seqs)  # 预测样本属于每个类别的概率
result_labels = np.argmax(result, axis=1)  # 获得最大概率对应的标签
y_predict = list(map(str, result_labels))
y_predict

y_predict=np.array(y_predict,dtype='int32')
y_predict.shape
y_predict
y_predictsize=y_predict.size
y_predict=y_predict.reshape(y_predictsize,1)
y_predict
y_test
print('准确率', sklearn.metrics.accuracy_score(y_test, y_predict))
print('平均 f1-score:', sklearn.metrics.f1_score(y_test, y_predict, average='weighted'))

model.save('model.h5')
```

问题二代码

```
import pandas as pd
import re
import numpy as np
```

```
import jieba
import sklearn
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans
from sklearn.externals import joblib
import codecs

sExcelFile="附件 3.xlsx"
df = pd.read_excel(sExcelFile,sheet_name='Sheet1')                    #读取 excel 文件
#liuyan=df['留言详情']                                                  #读取留
言
df['主题+留言']=df['留言主题']  +df['留言详情']
df['主题+留言']
df['热点号']=''
dfnew=df.copy()

area=['A1 区','A2 区','A3 区','A4 区','A5 区','A6 区','C1 区','C5 市','K2 区','G1 区','A7
县','A8 县','K3 县','J4 县','F6 县','F 市','D 市','A9 市','L 市','B 市','C 市']

len(area)

for i in range(df.shape[0]):
    for j in range(len(area)):
        if re.search(area[j], df['主题+留言'][i]):
            df['热点号'][i]=j

df['热点号']
df['热点号'].value_counts()
```

```
for i in range(df.shape[0]):
    if (df['热点号'][i]):
        dfnew.drop(i,inplace=True)


dfnew['热点号'].value_counts()
dfnew

liuyan=dfnew['主题+留言']
liuyan

jieba.load_userdict('newdic1.txt')
data_cut = liuyan.apply(lambda x: jieba.lcut(x))                    #分词
data_cut
stopWords = pd.read_csv('stopword.txt', encoding='GB18030', sep='hahaha', header=None)
stopWords = ['≮', '≯', '≠', '≮', ' ', '"', '月', '日', ' - ','\n','\t'] + list(stopWords.iloc[:, 0])
data_after_stop = data_cut.apply(lambda x: [i for i in x if i not in stopWords])
#去停用词
data_after_stop
adata = data_after_stop.apply(lambda x: ' '.join(x))
adata

adata.to_csv('./去停用词后.csv')


vectorizer = CountVectorizer()
# 该类会统计每个词语的 tf-idf 权值
transformer = TfidfTransformer()
```

```
# 第一个 fit_transform 是计算 tf-idf 第二个 fit_transform 是将文本转为词频矩阵
tfidf = transformer.fit_transform(vectorizer.fit_transform(adata))
 # 获取词袋模型中的所有词语
word = vectorizer.get_feature_names()
 # 将 tf-idf 矩阵抽取出来，元素 w[i][j]表示 j 词在 i 类文本中的 tf-idf 权重
weight = tfidf.toarray()

fileName = "TF-IDF_Result.txt"
result = codecs.open(fileName, 'w', 'utf-8')
for j in range(len(word)):
    result.write(word[j] + ' ')
result.write('\r\n\r\n')

    # 打印每类文本的 tf-idf 词语权重，第一个 for 遍历所有文本，第二个 for 便利某一类文本下的词语权重
for i in range(len(weight)):
    for j in range(len(word)):
        result.write(str(weight[i][j]) + ' ')
    result.write('\r\n\r\n')
result.close()

print('Start Kmeans:')

# 选择 7 个中心点
clf = KMeans(n_clusters=7)

# clf.fit(X)可以把数据输入到分类器里
clf.fit(weight)

# 7 个中心点
```

```
print( 'cluster_center:')
print(clf.cluster_centers_)

# 每个样本所属的簇
# print(clf.labels_)

i = 1
while i <= len(clf.labels_):
    dfnew['热点号'][i - 1] = (clf.labels_[i - 1])+21
    i = i + 1

df['热点号'].value_counts()
dfnew['热点号'].value_counts()

#df.combine_first(dfnew)
df['热点号']
dfnew['热点号']

for i in range(21):
    dfi= df.loc[df['热点号'] == i, :]
    name='./第二题/区分类'+str(i)+'.xlsx'
    dfi.to_excel(name)

for i in range(7):
    dfi = dfnew.loc[dfnew['热点号'] == (i+21), :]
    name = './第二题/A 市分类' + str(i+21) + '.xlsx'
    dfi.to_excel(name)


    # 用来评估簇的个数是否合适，距离越小说明簇分的越好，选取临界点的簇
```

个数

#print('inertia:')

#print(clf.inertia_)

保存模型

#joblib.dump(clf, 'km.pkl')

问题三代码

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from scipy.linalg import norm
import jieba

sExcelFile="附件 4.xlsx"
df = pd.read_excel(sExcelFile,sheet_name='Sheet1')                    #读取 excel 文件
#liuyan=df['留言详情']                                                  #读取留言
df['问答']=df['留言详情']  +df['答复意见']
df['相关性']=''

wenda=df['问答']
jieba.load_userdict('newdic1.txt')
data_cut = wenda.apply(lambda x: jieba.lcut(x))                       #分词
data_cut
stopWords = pd.read_csv('stopword.txt', encoding='GB18030', sep='hahaha', header=None)
stopWords = ['≮', '≯', '≠', '≮', ' ', '"', '月', '日', '–','\n','\t'] + list(stopWords.iloc[:, 0])
data_after_stop = data_cut.apply(lambda x: [i for i in x if i not in stopWords])
#去停用词
```

```
data_after_stop
adata = data_after_stop.apply(lambda x: ' '.join(x))
adata
adata.to_csv('./去停用词后.csv')


wen=df['留言详情']
data_cut = wen.apply(lambda x: jieba.lcut(x))                    #分词
data_after_stopw= data_cut.apply(lambda x: [i for i in x if i not in stopWords])
#去停用词
adataw = data_after_stopw.apply(lambda x: ' '.join(x))

da=df['答复意见']
data_cut = da.apply(lambda x: jieba.lcut(x))                    #分词
data_after_stopwd= data_cut.apply(lambda x: [i for i in x if i not in stopWords])
adatad = data_after_stopwd.apply(lambda x: ' '.join(x))



for i in range(df.shape[0]):
    cv = TfidfVectorizer(tokenizer=lambda s: s.split())
    corpus = [adataw[i],adatad[i]]
    vectors = cv.fit_transform(corpus).toarray()
    # 计算 TF 系数
    df['相关性'][i] =np.dot(vectors[0], vectors[1]) / (norm(vectors[0]) *
norm(vectors[1]))

df.to_excel('带相关性数据.xlsx')
```