“智慧政务”中的文本挖掘应用

摘要

本文针对各类社情民意相关的文本数据不断攀升，导致相关部门留言划分和热点整理的工作陷入瓶颈的现象，根据互联网公开渠道的数据，运用机器学习得出一套解决方案，并建立相关模型挖掘热点问题，并给出答复质量评价标准。

针对任务一，首先进行数据预处理。由于文本分类方法较多，故将**传统机器学习**与**深度机器学习**的方法进行对比训练模拟。传统机器学习先进行分词，构建词向量空间、TF-IDF 权重策略，然后从主流分类模型中得到最优模型——逻辑回归模型；深度机器学习采用卷积神经网络 CNN 模型，使用交叉熵损失函数，定义 Adagrad 优化算法；最后对两个模型进行 F-Score 评估得到的结果如下表所示：

模型	逻辑回归模型	CNN 模型
F-Score 评分值	0.90	0.93

针对任务二，先采用 **Levenshtein 算法**计算相似度，将结果大于 0.545 则作为同一类问题；再将热度评价指标定义为频数与点赞反对人数之和；接着根据 **FoolNLTK** 进行地点/人群的关键字识别，最后确定该热度问题的时间范围，部分结果如下表所示：

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	{'总热度': 2352, '提出人数': 6, '点赞': 2346, '反对': 0}	2019-01-14 至 2019-03-2	书记	请书记关注 A 市 A4 区 58 车贷案
……					

对应的热点问题详情表：

问题 ID	留言编号	留言用户	留言主题	留言时间	留言详情	点赞数	反对数
1	220711	A0003168 2	请书记关注 A 市 A4 区 58 车贷案	2019/2/21 18:45:14	尊敬的胡书记：您好!A4 区 p2p 公司 58 车贷...	821	0
……							

针对任务三，采用文本长度、关键词、完整性与及时性四个结构化特征进行质量评估，在利用潜在语义索引(LSI)计算答复与留言的**主题相似度**，利用主题相似度这个文本特征再和结构化特征评分融合，得出最终的答复质量评分。

关键词：自然语言处理，多文本分类，热点问题挖掘，答复质量评价，相似度算法，文本实体识别

目录

一、问题重述

1.1 背景及研究意义

“智慧政务”即通过“互联网+政务服务”构建智慧型政府，利用云计算、移动物联网、人工智能、数据挖掘、知识管理等技术，提高政府在办公、监管、服务、决策的智能水平，形成高效、敏捷、公开、便民的新型政府，实现由“电子政务”向“智慧政务”的转变。

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、 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。运用互联网、大数据等现代信息技术，不仅能加快推进部门间信息共享和业务协同，简化群众办事环节，还能提升政府行政效能、畅通政务服务渠道。

1.2 问题的相关信息

附件一给出留言内容分类三级标签的划分体系，一级到三级是层层递进的关系。一级标签分类共有 15 类，二级标签分类共有 103 类，三级分类标签共有 390 类。样例如表 1：

表 1 内容分类三级标签体系

一级分类	二级分类	三级分类
城乡建设	安全生产	事故处理
城乡建设	安全生产	安全生产管理
…	…	…

附件二给出：495 条留言信息记录，包含留言编号，留言用户，留言主题，留言时间，留言详情以及一级分类标签。具体结构如表 2：

表 2 附件二表结构

留言编号	留言用户	留言主题	留言时间	留言详情	一级分类
744	A089211	建议增加 A 小区快递柜	2019/10/18 14:44	我们是 A 小区居民...	交通运输

附件三给出：30 条留言信息记，包含留言编号，留言用户，留言主题，留言时间，留言详情、点赞数和反对数。具体结构如表 3：

表 3 附件三表结构

留言编号	留言用户	留言主题	留言时间	留言详情	点赞数	反对数
744	A089211	建议增加 A 小区快递柜	2019/10/18 14:44	我们是 A 小区居民...	100	2

附件四给出：120 条留言答复意见记录，包含包含留言编号，留言用户，留言主题，留言时间，留言详情、答复意见和答复时间。具体结构如表 4

表 4 附件四表结构

留言编号	留言用户	留言主题	留言时间	留言详情	答复意见	答复时间
744	A089211	建议增加 A 小区快递柜	2019/10/18 14:44	我们是 A 小区居民...	网民‘A089211’你好...	2019/10/19 8:40

1.3 问题重述

任务一：参考附件一提供的内容分类三级标签体系，根据附件二给出的留言信息，建立关于留言内容的一级标签分类模型，并使用 F-Score 对分类方法进行评价。

任务二：根据附件三将某一时段内反映特定地点或特定人群的热点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。

任务三：针对附件四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、模型的假设

假设一：回归是一种参数方法，模型为了分析需求对数据做出一定的假设都是合理的。

假设二：本文使用的卡方检验需要回归模型和分类问题两个变量是相互独立的且为正态分布。

假设三：附件三中的“留言主题”能完全准确的代表“留言详情”的内容，使分类结果更准确。

假设四：评价指标选取具有合理性、代表性

三、符号说明

符号	符号说明
N	某个单词在文档中的频次
M	文档的总单词数
D_w	出现了某单词的文档数量
A	卡方检验中的实际值
T	卡方检验的理论值
V	自由度
a	能检索到的分类标签正确的数量
c	未检测到的分类标签正确的数量

$Z(i,j)$	对应特征图的像素
t	特征图的通道数
y_i	经过卷积层得到的第i个特征数据
q_{ij}	在卷积层中第i个输入特征对应第j个输出特征数据的卷积核权值
W	全连接层的权重
θ_i	参数向量
g_t	目标函数对参数的梯度
S_i	排名第i的热点问题的点赞总人数
O_i	排名第i的热点问题的反对总人数

四、问题分析

4.1 对问题一的分析

该任务是需要根据附件二提供“留言主题”和“留言详情”的内容创建模型对该条留言进行一级标签分类，并且分类标签在附件一中有说明。因此，首先对数据进行预处理，将“留言主题”与“留言详情”进行合并，同时需要进行停用词处理，以便提高最终结果的准确率。

其次，由于该任务是一个典型的中文文本多分类问题，在使用数据挖掘分类方法的基础上，经过训练地标记实例模型，同时文本方法较多，所以我们选择了传统学习与深度学习的方法分别进行训练模拟，其中传统学习方法从数据预处理、中文分词、去除停用词、构建词向量空间、权重策略、选择分类模型、模型评估等过程进行处理分析；深度学习方法采用 CNN(Convolutional Neural Networks)模型。最后通过比较得出准确率较高的模型。

4.2 对问题二的分析

该任务既是一个分类问题也是一个统计问题，同时也包含了挖掘提取操作，是一个较为综合的题目。首先需要根据文本相似度对所有留言进行分类，其次根据留言数量、点赞数和反对数三者之和对问题进行排名，选出前 5 名作为热点问题，作为热点问题详情表；再利用 Foolnltk 算法提取每类问题的特定地点或人群，最后将每类问题点赞最多的作为问题描述并计算每类热点问题的时间段，作为热点问题表。

4.3 对问题三的分析

任务三是对相关部门答复意见进行评价总结，通过语义分析，从答复的相关性、完整性、可解释性等多个角度分别对答复意见作出评价，然后进行权重分配综合得出答复意见的质量评分，由此可得出一套答案质量评价方案。

首先根据答复长度，答复是否关键词、回答时间等指标进行初步判断是否直接、具体、及时的答复了留言，然后利用主题相似度给出答案的相关性评价。

4.4 整体结构图

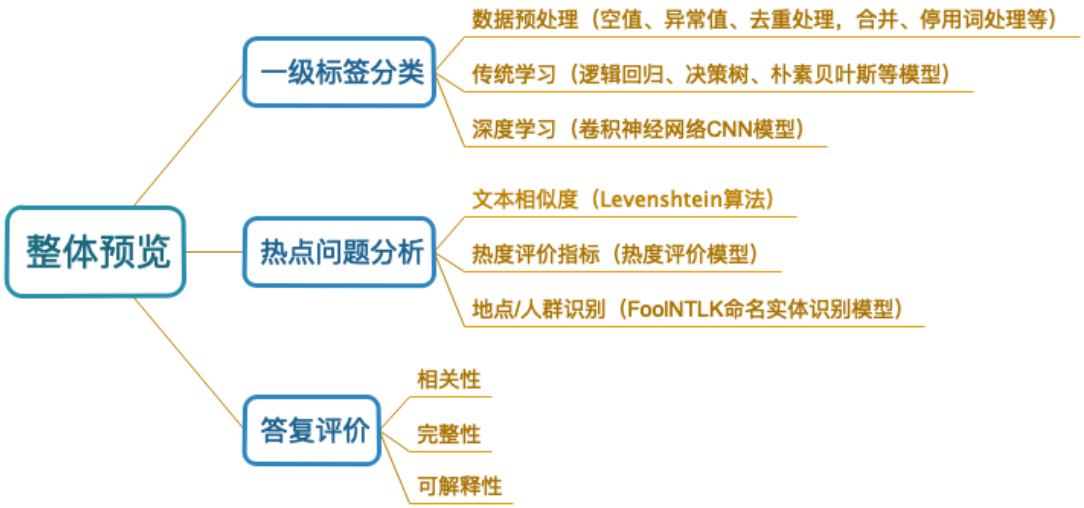


图 1　问题分析结构图

五、问题一模型的建立与求解

5.1 模型的建立

该问题是需要将群众留言按一级分类标签进行分类，而 python 文本分类的算法多种多样，故，我们将从基于传统机器学习（数据处理到模型预测）和基于深度学习（CNN 模型）两个方面对文本进行分类，最后通过比较准确率得出最终的分类结果。

5.1.1 数据预处理

源数据常常会有空值、重复值、无意义的字符或格式不符合要求等等问题会对机器学习产生影响，所以在进行实验前要数据清洗进行文本降噪。本实验对源数据进行填补空值、去重、选择文本范围、去除无用字符等处理。

1）为了方便处理将文本分类标签数字化，即

$$N_{i+1} \to i++(i=0,1,\cdots,14)$$

其中 N_{i+1} 表示第 $i+1$ 个文本标签；

2）为了使结果准确率更高，将留言主题与留言详情进行文本融合处理；中文表达中最常用的功能性词语是限定词，如“的”、“一个”、“这”、“那”等。这些词语的使用较大的作用仅仅是协助一些文本的名词描述和概念表达，并没有太

多的实际含义，对分类也没有加载所以应该除去这些词。所以需要利用相关停用词将无关文本进行过滤。本实验采用将哈工大停用词、四川大学机器智能实验室停用词、百度停用词、中文停用几个合并为一个的停用词表，去掉停用词表中的所有停用词。

5.1.2 传统学习[2]

使用传统机器学习方法解决文档分类问题一般分为：数据预处理、中文分词、去除停用词、构建词向量空间、权重策略、选择分类模型、模型评估等过程。本次实验将按照传统机器学习各个步骤完整地进行处理。

5.1.1.1 分词处理

在对数据进行部分预处理之后（停用词处理在分词处理值之后），采用结巴分词库[8]对文本进行分词。结巴分词支持三种分词模式：全模式、精确模式和搜索引擎模型，此外，结巴分词还支持繁体分词和自定义词典。结巴分词的系统框架如下图所示：

图 2 结巴分词的系统框架

其中涉及的算法有：

- 基于前缀词典实现词图扫描，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，采用 Viterbi 算法进行计算；
- 基于 Viterbi 算法的词性标注；
- 分别基于 TF-IDF 和 Textrank 模型抽取关键词；

5.1.1.2 词频统计与词云图

源数据每个类别的数据多少不一定，各类别的数据量不同影响分类效率，所以要提前观察数据了解数据，选取合适的数据量。

故，统计每个类别的词频数，及每个词的频数形成统计图与词云图，便于直观观察数据，选取数据。

5.1.1.3 构建词向量空间

文本的向量表示

机器理解不了中文，所以要将文本进行词向量处理，将自然语言中的字词转为计算机可以理解的稠密向量，如 One-Hot 表达。下面以一个例子来说明词向量空间的构建过程。

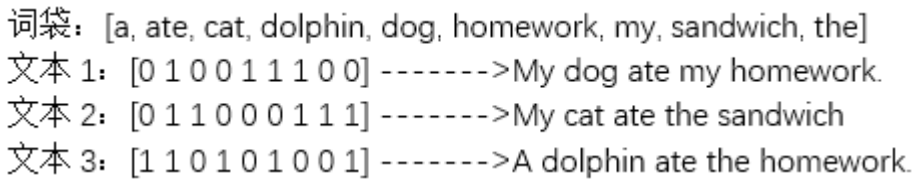


图 3 构建词向量

其中，One-Hot 表达是指从非结构化数据到结构化数据转化，将每个词表示为一个长长的向量。词袋是指所有词的不重复构成，最终将文本转化为词向量矩阵。

但是忽略了词频信息并且也有句子长度不一致问题，所以还需要根据 TF－IDF 权重策略增加词频信息与归一化操作。

TF-IDF 权重策略

权重策略文档中的高频词应具有表征此文档较高的权重，除非该词也是高文档频率词。

TF(Term frequency)即关键词词频，是指一篇文档中关键词出现的频率。公式如下：

$$TF = N / M$$

其中，N 表示单词在某文档中的频次，M 表示该文档的总单词数。

IDF(Inverse document frequency)指逆向文本频率，是用于衡量关键词权重的指数，公式如下：

$$IDF = \log\left(\frac{D}{D_w}\right)$$

其中，D 表示总文档数，D_w 表示出现了该单词的文档数。

最终，TF-IDF 权重策略公式如下所示：

$$TF - IDF = TF \times IDF$$

7.1.2.1 也应用改原理，转到

5.1.1.4 特征选择

特征选择是降维的一种方式，就是试图去减少特征数据集中的属性(或者称为特征)的数目，比较有名的特征选择有过滤法（Filter），包裹法(Wrapper)，嵌入法(Embedded)。本实验将采用过滤法中的卡方检验的方法选择特征。

卡方检验是数理统计中一种常用的检验两个变量独立性的方法，对于回归和

分类问题常用卡方检验等方式对特征进行测试。以下是卡方检验的计算公式:

$$x^2 = \sum \frac{(A-T)^2}{T}$$

其中，A是实际值，T为理论值，x^2的值表示衡量理论与实际的差异程度。

根据自由度查询卡方分布的临界值γ（见附录图 12），将计算的值与临界值比较。自由度公式为：

$$V = (H-1)\cdot(C-1)$$

其中，H表示行数，C表示列数；如果$x^2 < \gamma$，则假设成立；反之，则不成立。

5.1.1.5 模型选择与调参

文本分类方法模型主要分为四个大类：基于规则的分类模型、基于概率统计的模型、基于几何的模型、基于统计的模型。

1) 基于规则使用许多条规则来表述类别，类别规则可以通过领域专家定义，也可以通过计算机学习获得。优点就是时间复杂度低、运算速度快。典型模型决策树。

2) 基于概率的模型：假设未标注文档为d，类别集合为$C = \{c_1, c_2, \cdots, c_m\}$，概率模型分类是对$1 \le i \le n$求条件概率模型$P(c_i \mid d)$，将与文档$d$条件概率最大的那个类别作为该文档的输出类别。典型模型朴素贝叶斯分类器[5]

3) 基于几何的模型使用向量空间模型表示文本，文本就被表示为一个多维的向量，那么它就是多维空间的一个点。通过几何学原理构建一个超平面将不属于同一个类别的文本区分开。典型模型支持向量机(SVM)。

4) 基于统计的模型：事实上无论是朴素贝叶斯分类模型，还是支持向量机分类模型，也都采用了统计的方式。文本分类算法中最典型的基于统计的分类模型就是 k 近邻模型[3]。

本实验将采用各种分类器进行分类评估选择分数最高的分类器进行调参，得到最优模型。

5.1.2 深度学习——CNN 模型[4]

文本分类的关键在于准确提炼文档或者句子的中心思想，而提炼中心思想的方法是抽取文档或句子的关键词作为特征，基于这些特征去训练分类器并分类。而 CNN(Convolutional Neural Networks)的卷积和池化过程就是一个抽取特征的过程，当我们可以准确抽取关键词的特征时，就能准确的提炼出文档或句子的中心思想。因此我们利用 CNN 模型进行文本分类。

5.1.2.1 数据处理

1.创建数据集和数据字典

数据集会对应一个数据集字典，即将文本中每一个字映射到字典中得到一个数字 ID，作为之后模型输入的依据。

将文本转化为序列表示，即数据是以数字 ID 的方式表示一个句子。所以每个句子都是以一串整数来表示的，每个数字都是对应一个字。

2.创建数据读取器 train_reader 和 test_reader

在需要遍历数千或数百万个数据行时，可以使用快速的顺序读取器(一次读取结果集中的数行数据)，数据读取器会以一种非常有效的方式完成此项工作。

通过多线程方式，通过用户自定义的映射器 mapper 来映射 reader 返回的样本（到输出队列)。

5.1.2.2 配置网络

1.定义网络

定义 CNN 网络结构为：输入层→卷积层→池化层→全连接层，如下图所示：

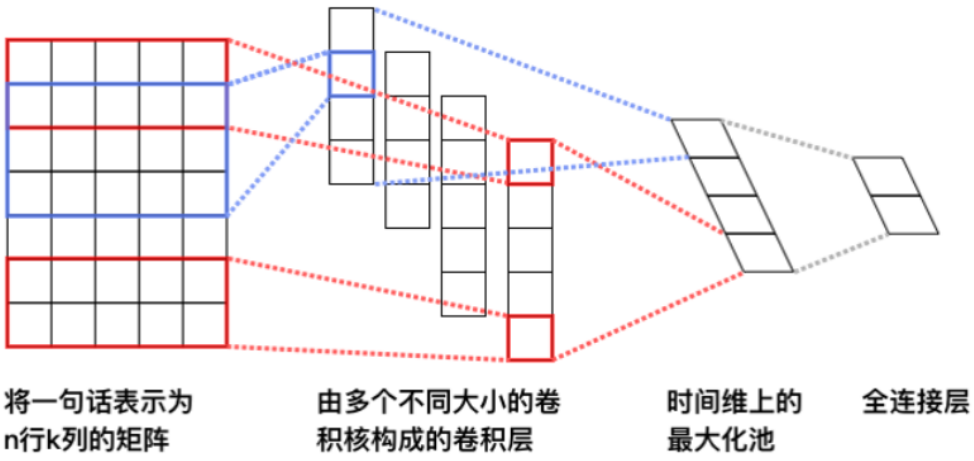


图 4 CNN 网络结构

输入层：

卷积神经网络的输入层可以处理多维数据，在该任务中使用二维卷积神经网络的输入层接受的是矩阵（即二维数组），将一句话表示为$n*k$的矩阵。

卷积层：

卷积层的功能是对输入数据进行特征提取，其内部包含多个可学习的卷积核，组成卷积核的每个元素都对应一个权重系数和一个偏差量，类似于一个前馈神经网络的神经元。卷积核在工作时对输入特征做矩阵元素乘法求和并叠加偏差量为：

$$Z^{l+1}(i,j)=\left[Z^{l}\otimes w^{l+1}\right](i,j)+b=\sum_{t=1}^{t_l}\sum_{x=1}^{f}\sum_{y=1}^{f}\left[Z_t^{l}(s_0 i+x,s_0 j+y)w_t^{l+1}(x,y)\right]$$

$$(i,j)\in\{0,1,\cdots,L_{l+1}\}$$

$$L_{l+1}=\frac{L_l+2p-f}{s_0}+1$$

其中，b为偏差量，Z^{l}和Z^{l+1}表示第$l+1$层的卷积输入和输出，也被称为特征图，L_{l+1}为Z^{l+1}的尺寸，$Z(i,j)$对应特征图的像素，t为特征图的通道数，f、s_0和p是卷积层参数，对应卷积核大小、卷积步长和填充层数。

卷积层用若干个卷积核与词向量矩阵进行卷积，可以设计宽度和词向量维度相同、高度变化的卷积核，在词语序列方向上进行卷积操作，形式如下所示：

$$y_i=g\left(\sum_{i\in D}x_i\cdot q_{ij}+b_j\right)$$

其中，y_i表示经过卷积层得到的第j个特征数据，$g()$为激活函数，x_i表示输入数据的集合 D 中第 i 个输入数据，q_{ij}表示在卷积层中第 i 个输入特征对应第 j 个输出特征数据的卷积核权值。

本任务中一个句子是由多个词拼接而成的，如果一个句子有 n 个词，且第 i 个词表示为x_i，词x_i通过 embedding 后表示为k维的向量，即$x_i\in\Re^k$，则一个句子可以形式化如下：

$$X_{1:n}=x_1\oplus x_2\oplus\cdots\oplus x_n$$

一个包含 h 个的词的词窗口表示为：

$$X_{i:i+h-1}\in\Re^{hk}$$

一个 filter 是大小为$h*k$的矩阵转换成的长度为$h*k$一维向量，W 为全连接层的权重，表示为:

$$W \in \Re^{hk}$$

通过一个 filter 作用一个词窗口提取可以提取一个特征c，如下：

$$c_i = g(W \cdot X_{i:i+h-1} + b)$$

$$c = [c_1, c_2, \cdots, c_{n-h+1}]$$

池化层：

在卷积层进行特征提取后，输出的特征图会被传递至池化层进行特征选择和信息过滤。池化层包含预设定的池化函数，其功能是将特征图中单个点的结果替换为其相邻区域的特征图统计量。该任务对特征图采用时间维度上的最大池化操作得到此卷积核对应的整句话的特征。池化模型的一般表示形式为：

$$A_k^l(i,j) = \left[\sum_{x=1}^{f} \sum_{y=1}^{f} A_k^l(s_0 i + x, s_0 j + y)^p \right]^{\frac{1}{p}}$$

其中，p 是预指定参数，当 $p \to \infty$ 时，池化在区域内取极大值，被称为极大池化。

本任务中的池化操作是对一个 filter 提取到的 feature map 进行 max pooling，得到 $\hat{c} \in \Re$ 即：

$$\hat{c} = \max(c)$$

若有 m 个 filter，则通过一层卷积、一层池化后可以得到一个长度为 m 的向量 $z \in \Re^m$：

$$z = [\hat{c}_1, \hat{c}_2, \cdots, \hat{c}_m]$$

再使用 softmax（归一化指数函数）将输出的结果进行处理，使其多个分类的预测值 $P(i)$ 的范围在 $[0,1]$ 之间，为后续用交叉熵来计算损失做准备，softmax 函数如下所示：

$$P(i) = \frac{\exp(\theta_i^T x)}{\sum_{K=1}^{K} \exp(\theta_i^T x)}$$

其中 θ_i 和 x 是列向量。

全连接层：

将所有卷积核得到的特征拼接起来即为文本的定长向量表示，将其连接构建出完整的模型。

本任务将向量 z 输入到全连接层，得到最终的特征提取向量 y（注意与 filter 进行区分）：

$$y = W \cdot z + b$$

2.定义损失函数

本任务中使用交叉熵损失函数，交叉熵能够衡量同一个随机变量中的两个不同概率分布的差异程度，在机器学习中就表示为真实概率分布与预测概率分布之间的差异。交叉熵的值越小，模型预测效果就越好。表达式如下

$$L = -\sum_{c=1}^{M} R_c \log(P_c)$$

其中 M 类别的数量，R_c 表示变量，正确为 1，否则为 0，P_c :预测正确的概率

3.定义优化算法

本任务使用的是 Adagrad 优化方法，Adagrad 优化方法能够在训练中自动对学习率进行调整，对于出现频率较低参数采用较大的学习率 α 更新；相反，对于出现频率较高的参数采用较小的学习率 α 更新，多用于处理稀疏数据。

学习率 α 的会随着每次迭代而根据历史梯度的变化而变化，表达式如下：

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\varepsilon I + diag(G_t)}} \cdot g_t$$

其中，t 表示某次迭代时刻，θ_t 表示参数向量，g_t 表示目标函数对参数的梯度，G_t 表示一个对角矩阵，ε 表示平滑项，防止除零操作。

5.1.3 分类模型的评估

常用的评估分类系统性能的标准：召回率(Recall)、准确率(Precision)、F1-评测值(F1-measure)、微平均(Micro-average)和宏平均(Macro-average)。另外一些使用较少的评估方法包括平衡点(break-even point)、11 点平均正确率(11-point average precision)等。

本实验将采用召回率(Recall)、准确率(Precision)、F-Score 还有混淆矩阵进行评估。

1) 召回率(又称查全率)，定义如下式：

$$\mathrm{Re}call_i = \frac{a}{a+c} \times 100\%$$

其中，a 表示能检索到的分类标签正确的数量，c 表示未检测到的分类标签正确的数量。

2) 准确率(又称查准率)，定义如下式：

$$\mathrm{Pr}ecision_i = \frac{a}{a+b} \times 100\%$$

其中， b 表示检测到的分类标签不正确的数量。

3) F1-评测值，定义如下式：

$$F_{1i}=\frac{1}{N}\sum_{i=1}^{N}\frac{2\times \mathrm{Pr}ecision_i\times \mathrm{Re}call_i}{\mathrm{Pr}ecision_i+\mathrm{Re}call_i}$$

4) 混淆矩阵也称误差矩阵，是表示精度评价的一种标准格式，用 n 行 n 列的矩阵形式来表示。混淆矩阵的每一列代表了预测类别，每一列的总数表示预测为该类别的数据的数目；每一行代表了数据的真实归属类别，每一行的数据总数表示该类别的数据实例的数目。

表 5 混淆矩阵

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)

5.2 模型的求解

5.2.1 求解过程

传统机器学习：

step1.探索数据：查看每类留言条数、词频、词云图等信息。

step2.数据预处理：进行数据选取、格式化，jieba 分词等加工。

step3.数据向量化：利用 CountVectorizer，TfidfTransformer 两个模块将数据转化为 tf-idf 权重训练词频向量。

step4.特征提取：利用卡方验证进行特征选取(初设 2500)，之后进行调优。

step5.模型训练：选择准确率最高的分类模型进行训练并利用 GridSearchCV 进行参数调优。

step6.模型预测：选择最好的模型与参数进行预测并保存预测结果。

step7.模型评估：查看准确率，召回率，F-Score，混淆矩阵。

step8.查看错误记录：查看具体分类错误的记录，便于后期改进学习。

深度机器学习：

step1.数据预处理：进行数据选取、格式化等加工。

step2.创建数据字典：将文字转化为数字后将文本映射成数字序列。

step3.创建 CNN 网络：获取分类器，创建交叉熵损失函数和准确率，并定义 Adagrad 优化方法。

Step4.模型训练：通过迭代进行模型训练，调整迭代次数选择最佳模型，得到其最高准确率。

Step5.模型评估：查看准确率，召回率，F-Score。

5.2.1 过程结果

1. 分词后得到的词频统计结果如下图所示：

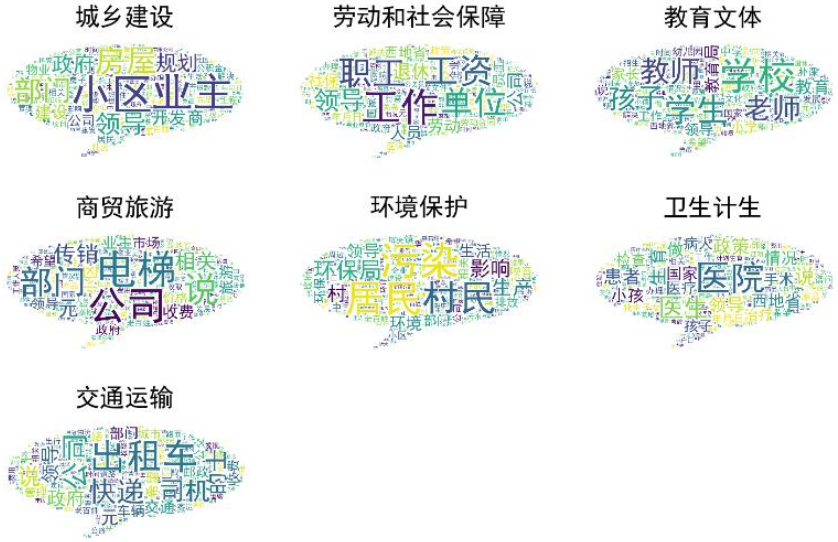


图 5 词频统计

2. 通过分类模型的结果分析比较得出传统方法的最优模型——逻辑回归模型，模拟的具体结果（评分值由高向低排列）如下表所示：

表 6 分类模型评分值

分类模型	评分值
逻辑回归模型	0.8728571428571429
高斯贝叶斯模型	0.8728142857142857
朴素贝叶斯模型	0.7875714285714286
K-最邻近模型	0.6185714285714285
决策树模型	0.48
随机森林	0.28
支持向量机模型	0.13842857142857143

3. 在分类模型进行评分之后得到理想模型，我们又对逻辑回归模型进行参数与卡方的 k 值调优处理，最后得到的最好的模型与 k 值如下图所示：

```
LogisticRegression(C=10, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=10, multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
2500 : 0.8971428571428571
10000 : 0.9014285714285715
15000 : 0.9057142857142857
20000 : 0.9028571428571428
```

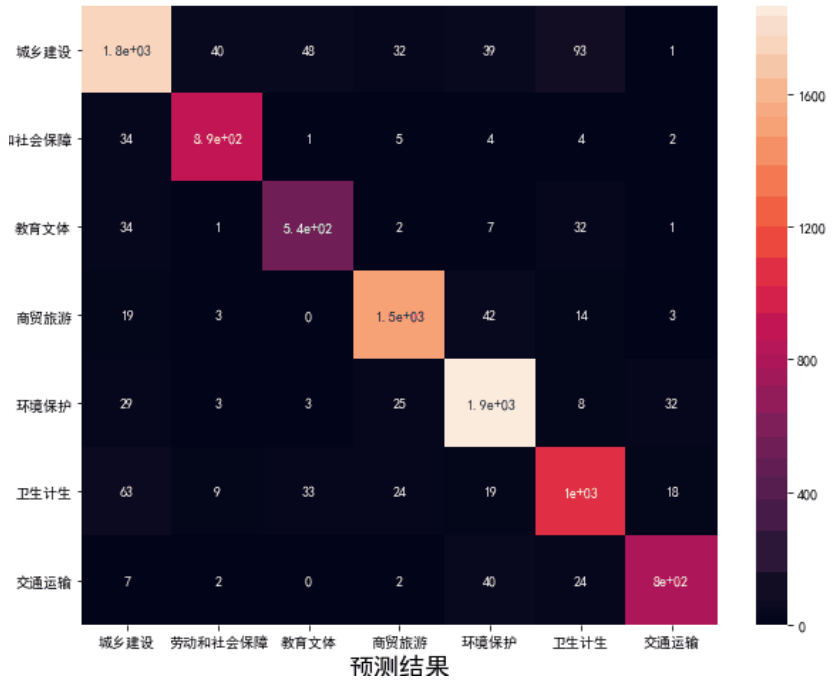
图 6 调优结果

4. 最终得到传统学习方法和深度学习方法得到的准确率如下表所示：

表 7 评分值

模型 \ 标签		城乡建设	劳动和社会保障	教育文本	商贸旅游	环境保护	卫生计生	交通运输	综合得分
逻辑回归模型	准确率	0.85	0.95	0.91	0.90	0.90	0.90	0.93	0.90
	召回率	0.79	0.92	0.93	0.84	0.99	0.96	0.89	
	F-Score	0.82	0.94	0.92	0.87	0.94	0.93	0.91	
CNN 模型	准确率	0.90	0.95	0.97	0.87	0.96	0.94	0.92	0.93
	召回率	0.92	0.95	0.95	0.90	0.94	0.92	0.88	
	F-Score	0.91	0.95	0.96	0.89	0.95	0.93	0.90	

5. 两种模型的混淆矩阵如下所示：



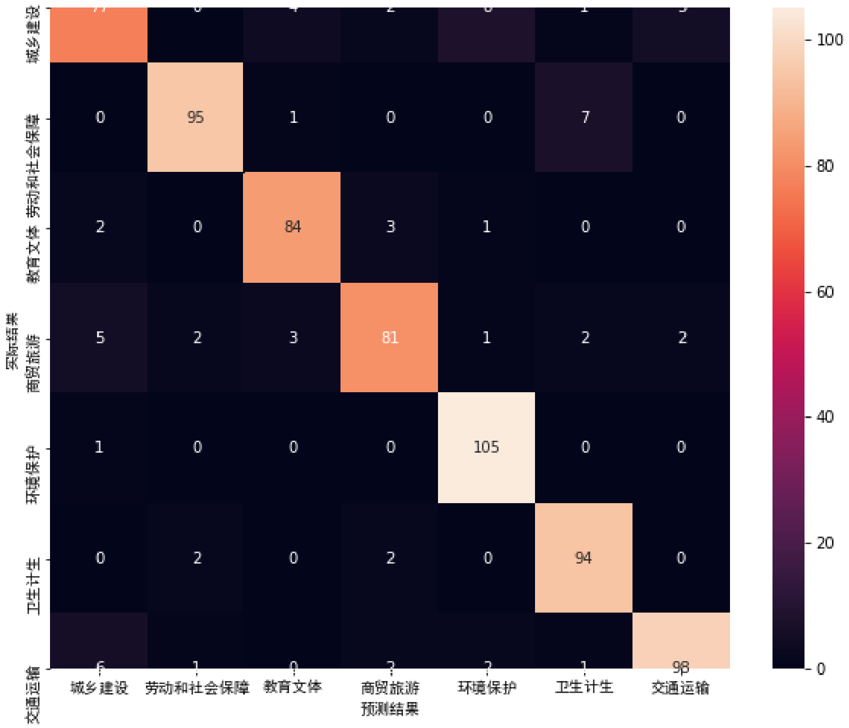


图 7 混淆矩阵（上：CNN，下：回归模型）

6. 查看分类错误的具体留言，部分结果如下：

```
【城乡建设】 预测为 【环境保护】: 8 例.

312     四季 美景 栋 号 餐饮 油烟 扰民 多年 市区 四季 美景 栋 号 正宗 老字号 牛肉 粉...
603     大坪 工业园 土方 施工人员 无证 施工 工地 安全员 大坪 工业园 酿溪镇 雷家坳 村 工...
1936    临县 新安镇 生活 用水 质量 合格 临县 新安镇 居民 生活 用水 合格 经常出现 水 浑...
1949    县岩 嘴 黄岭 岗路 挖 自来水管 挖出 黄金 领导 \r 岩嘴 黄岭 岗路 挖 自来水管 ...
629     市陶家 十字路口 扰民 整改 到位 前次 陶家冲 十字路口 扰民 点点 好转 死灰复燃 摆摊...
326     绿城 桂花城 小区 西头 马路 摆摊 噪音 急需解决 尊敬 肖 局长 住 大道 绿城 桂花城...
22      杜鹃 文苑 小区 外 非法 汽车 检测站 开业 市政府 交警支队 安监局 环保局 区政府 市...
924     江镇 城南 新 市场 油烟 噪声 扰民 住 江镇 城南 新 市场 大道 号 三禾 烧烤店 一...
dtype: object
```

	留言主题	留言详情	一级标签	lab_id
312	A4区四季美景12栋82号餐饮油烟扰民多年	\n\t\t\t\t\n\t\t\t\t	A市A4区四季美景12栋82号正宗老字号	城乡建设	1

```
【劳动和社会保障】 预测为 【卫生计生】: 7 例.

6438    咨询 生育 津贴 事宜 合法 夫妻 二胎 孕妇 误服 药物 实施 人流 缴纳 医保 生育 申...
6010    乡镇 卫生院 医生 职绩 数 上报 难 尊敬 胡 厅长 乡镇 卫生院 医生 参加 农卫 副 ...
5184    市村医 养老 未 解决 陈书记 你好 村医 年纪 卫生局 对村 卫生室 给予 新农 合 报销...
5383    中专生 户口 请教 局长 问 麻烦 告诉 信息工程 学样 级 中专 毕业生 读书 时 户口 ...
6615    询问 省 医保 社区 医院 狂犬 疫苗 注射 夫人 猫 咬伤 学士 街道 卫生 服务中心 注...
6148    资格考试 报名表 打印 咨询电话 接通 市人 社局 考区 考生 年度 一级 建造师 含 专业...
5724    西湖 未入 旧 农保 村民 申请 旧农保 尊敬 张局 西湖 未入 旧 农保 村民 申请 旧农...
dtype: object
```

	留言主题	留言详情	一级标签	lab_id
6438	咨询K市生育津贴有关事宜 \n \n	合法夫妻，二胎因孕妇误服药物而不得不实施人流，由于一直都有缴...	劳动和社会保障	2

图 8 分类错误的留言

5.3 结果分析

1. 我们可以从图 5 中发现城乡建设的关键词主要是小区、业主、规划、领导等；劳动和社会保障的关键词主要是工作、公司、职工、单位、领导等；商贸旅游的关键词主要是公司、电梯、部门等；交通运输的关键词主要是出租车、快递、公司等。可以看出在城乡建设、劳动和社会保障中都同时出现关键词“领导”；

劳动与社会保障、商贸旅游和交通运输中出现“公司”，所以这就有可能造成标签判别失误;因此,我们需要多种模型进行分析比较得出准确率最高的一种模型。

2.根据表 7 我们可以看到深度学习的准确率比传统学习的评估分数高,可能因为深度学习没有平均选取每类数据，训练集大，所以准确率高。且从实验整体情况来看，CNN 模型的分值较平均而且每次分值稳定，而回归模型的分值最高和最低值的差异较大，而且传统学习需要大量的人工操作，所以在任务一中的留言分类中选择深度学习的 CNN 模型。

3. 根据表 7 我们可以看到综合来说”城乡建设”和”商贸旅游”的问题分类性能较差，均在 0.9 以下；从混淆矩阵可以看到其中错误率最高的是”城乡建设”预测为”教育文体”有 8 条,其次是”劳社保障”预测为”卫生计生”有 7 条。查看错误具体记录可以看到错误的”城乡建设”分类的留言基本是出现了地名、与环境保护有关的场所、景物名称等混淆词语，错误的”劳社保障”分类的留言基本是出现了生育类、卫生类名词、医院相关场所等混淆词语，之后的模型深入改进可以从这方面入手，进一步进行数据处理，提高准确率

六、问题二模型的建立与求解

6.1 模型的建立

该任务需要将相似度较高的热点问题（某一时段内群众集中反映的某一问题）的前 5 名整理成一张表格，使相关部门进行有针对性地处理，提高服务效率；并定义热点问题的热度评价指标，给出评价结果，将每个热点问题的留言详情结果以表格的形式展示。

6.1.1 热点问题分类——Levenshtein 算法[1]

Levenshtein.jaro 是计算 2 个字符串之间相似度的一种算法。主要用于数据连接（重复记录）方面的领域，该算法最后得分越高说明相似度越大，适合于较短的字符之间计算相似度。0 分表示没有任何相似度，1 分则代表完全匹配。该算法的距离公式如下所示：

$$d_j=\begin{cases}0 & if\,(m=0)\\ \frac{1}{3}\left(\frac{m}{|s_1|}+\frac{m}{|s_2|}+\frac{m-t}{m}\right) & otherwise\end{cases}$$

式中，s_1、s_2是需要对比的两个字符串，m 是匹配的字符数，t 是换位的数目。当s_1、s_2中字符相匹配但是字符位置不一样时发生换位操作，而上式中的 t 为不同顺序的匹配字符的数目的一半。

字符串 s_1 与字符串 s_2 在做匹配计算时，当两个字符的距离不大于匹配窗口的值即认为是匹配的。匹配窗口计算公式如下：

$$MV = \left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1$$

该算法给予了起始部分就相同的字符串更高的分数，它定义了一个前缀范围 P，对于要匹配的两个字符串，如果前缀部分有长度为 L 的部分字符串相同，则两字符串之间的距离为:

$$d_w = d_j + L \cdot P(1 - d_j)$$

其中，L 是前缀部分匹配的长度，P 是一个范围因子常量，用来调整前缀匹配的权值，但是 P 的值不能超过 0.25，因为这样最后得分可能超过 1。标准默认设置值 $P = 0.1$。

对应于该任务，我们将 $d_w > 0.545$ 的留言主题划分为同一热点问题，并将“点赞数”和“反对数”纳入考虑，最终得出排名前 5 的热点问题。

6.1.2 热度评价模型

我们将热度评价指标定义为支持该条热点问题的人数与反对人数之和，公式如下：

$$Q_i = N_i + S_i + O_i$$

其中，N_i 表示排名第 i 的热点问题的留言总人数；S_i 表示排名第 i 的热点问题的点赞总人数；O_i 表示排名第 i 的热点问题的反对总人数；

6.1.3 命名实体识别模型[7]——FoolNLTK

FoolNLTK 是一个使用双向 LSTM 构建的便捷的中文处理工具包，该工具不仅可以实现分词、词性标注和命名实体识别，同时还能使用用户自定义字典加强分词的效果。FoolNLTK 模块相比较于其他相似度模块使用简单，处理效率迅速，准确率还不错。所以在该任务中，使用该中文处理工具包进行命名实体识别。

基础的 RNN 网络实现了对当前时刻的输入和前一时刻状态的抽象表示，在预测状态时虽然能考虑到上一时刻的状态，但是在上下文距离过长情况下，容易产生梯度爆炸或梯度消失的情况，无法进行参数更新。LSTM 网络改变了 RNN 网络细胞元结构，通过门控机制，在处理长距离依赖的序列时有着较好的效果，它能够记忆先前的信息并应用到当前输出的计算中。LSTM 网络改变了 RNN 网络细胞元结构，通过门控机制，在处理长距离依赖的序列时有着较好的效果，它能够记忆先前的信息并应用到当前输出的计算中。

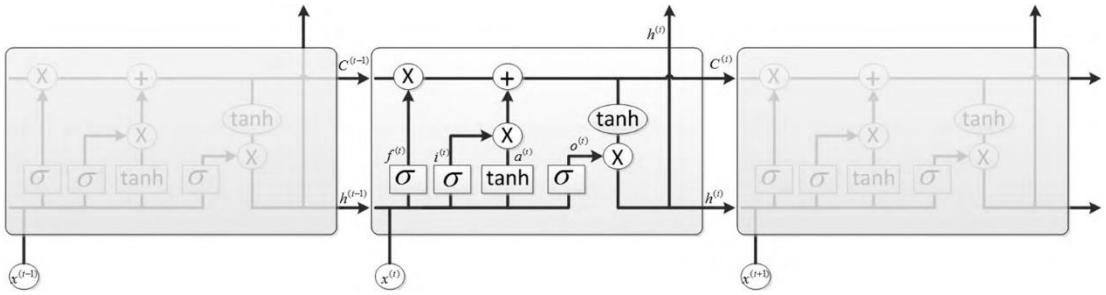


图 9 LSTM 网络示意图

一个 LSTM 细胞结构如图 1 所示，它包括了遗忘门、输入门、输出门 3 个部分，数据流经过这 3 个部分的处理，输出更新后的状态。这 3 个部分的作用机理如下：

遗忘门：

$$f^{(t)} = sigmoid(U_f x^{(t)} + W_f h^{(t-1)} + b_f)$$

输入门：

$$i^{(t)} = sigmoid(U_i x^{(t)} + W_i h^{(t-1)} + b_i)$$

输出门：

$$a^{(t)} = \tanh(U_a x^{(t)} + W_a h^{(t-1)} + b_a)$$

具体而言，FoolNLTK 采用 BiLSTM-CRF 模型，该模型集成了 LSTM 和 CRF 的优点，既能够捕获文本的全文特征，又能够在输出预测时考虑到标签的顺序性。

6.2 模型的求解

6.2.1 求解过程

step1.数据探索与预处理：进行数据选取、格式化，jieba 分词等加工

step2.相似度算法选择：选取几条数据测试各种距离算法与本实验的合适度，选择最佳算法与相似度阈值

step3.问题分类：利用 Levenshtein.jaro 计算相似度进行留言问题归类

step4.统计热点问题：根据每类问题的频数与点赞、反对数之和统计出前五名的问题作为热点问题

step5.特定地点/人群提取：利用 FoolNLTK 模块进行实体识别，并将频数最高的实体作为特定地点/人群

step6.其他信息：将点赞数最高的留言主题作为本类问题的描述，并计算此类问题时间段

step7.保存实验结果：整理合并实验结果，并按要求保存

6.2.2 过程结果

1. 相似度算法与阈值选择的结果如下表：

表 8 相似度计算

问题 1	问题 2	编辑距离	莱文斯坦比	jaro 距离	Jaro–Winkler 距离
五矿万境负一楼面积缩水	区圭塘路五矿万境水岸路段拥堵及执法不为	15	0.33	0.57	0.57
书记关注市区车贷案	严惩市车贷特大集资诈骗案保护伞	14	0.32	0.556	0.556
劳动东路魅力之城油烟扰民	魅力之城一楼装修违规严重污染空气	16	0.28	0.53	0.53
……					

2. 根据 Levenshtein 算法得出热点问题分类的留言明细如下表所示：

表 9 热点问题留言明细表

问题 ID	留言编号	留言用户	留言主题	留言时间	留言详情	点赞数	反对数
1	220711	A00031682	请书记关注 A 市 A4 区 58 车贷案	2019/2/21 18:45:14	尊敬的胡书记：您好!A4 区 p2p 公司 58 车贷...	0	821
1	194343	A000106161	A 市 58 车贷案警官应跟进关注留言	2019/3/1 22:12:30	您好！58 车贷案发，引发受害人举报投...	0	733
……							
……							
5	272089	A00061602	关于 A6 区月亮岛路 110kv 高压线的建议	2019/3/26 10:17:31	联名信——坚决要求 A 市润和又一城、三润城、润…	0	10
5	268250	A00072424	关于 A6 区月亮岛路沿线架设 110KV 高压电线杆的投诉	2019/3/26 10:17:31	尊敬的胡书记，您好！根据区政府和街道办及电力...	2	55
……							

2. 将表 9 的内容进行时间划分、热度评价指标和地点/人物提取得到排名前 5 的热点问题表，如下所示：

表 10 热点问题表

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	{'总热度': 2352, '提出人数': 6, '点赞': 2346, '反对': 0}	2019-01-14 至 2019-03-2	书记	请书记关注 A 市 A4 区 58 车贷案
2	2	{'总热度': 2113, '提出人数': 7, '点赞': 2106, '反对': 0}	2019-01-15 至 2019-11-11	A 市五矿万境 K9 县	A 市 A5 区汇金路五矿万境 K9 县存在一系列问题
3	3	{'总热度': 704, '提出人数': 6, '点赞': 698, '反对': 0}	2019-01-30 至 2019-09-16	绿地海外滩小区	A4 区绿地海外滩小区距长赣高铁最近只有 30 米不到，合理吗？
4	4	{'总热度': 252, '提出人数': 5, '点赞': 246, '反对': 1}	2019-04-29 至 2019-12-23	华悦城物业	A 市富绿物业丽发新城强行断业主家水
5	5	{'总热度': 182, '提出人数': 7, '点赞': 173, '反对': 2}	2019-03-26 至 2019-04-12	A6 区月亮岛路	关于 A6 区月亮岛路沿线架设 110kv 高压线杆的投诉

6.3 结果分析

1. 我们可以从表 8 中得到编辑距离没有归一化处理，无法确定阈值，莱文斯坦比只能计算文本是否一样没有考虑语义，而最适合本实验的相似度算法为 jaro 距离；不同类别的阈值基本小于 0.5，为了消除某些留言主题过短而产生的误差，所以根据多次调试将阈值为 0.545 效果最好。

2. 我们可以从表 9 中看到问题归类的结果中，只有类别 1 的有 1 条错误分类，其他问题没有出现分类错误的情况，说明模型效果较好。具体分析每类问题发现出现同一个地方但不同的问题被分为一类的情况，但本实验要求就是根据某一特定地点/人群，所以这个问题不严重，但在以后深入改进模型时可以考虑。

3. 我们可以从表 10 中看到热点问题的统计结果，可以看到 5 个热度最高的问题，模型效果符合预期，但热度评价指数只有留言频数与点赞、反对人数之和这一评价指标，可以在之后的改进中继续深入挖掘其他指标

七、问题三模型的建立与求解

7.1 模型的建立

答复意见质量评价对理解留言群众的信息需求、提高相关部门答复质量有重要作用。由于群众留言文本叙述方式的多样性，还有信息需求的复杂性和模糊性，如何判断相关部门的答复是否满足用户的需求是一个极其复杂的问题。答复意见质量评价是一个包含信息质量和自然语言处理技术的跨学科问题，答复的文本内容与非文本特征都可以作为评价其质量的依据。文本内容特征对质量的影响高于

非文本特征的影响,其中内容的完整性、可靠性、准确性以及群众的表达方式与答复质量都有着密切的关系，而高频词和答复的长度对答复质量的影响较小。

在本文中，结构化特征可以选取以下几个方面：

- 文本长度：群众对答复最直观的感受是文本长度,文本长度与信息量正相关，在本文中答复长度决定了是否直接回答问题还是转交至别的部门。
- 关键词：根据分析本文的回复格式得出具体回复都会含有“回复如下”、“答复如下”、“专此回复”、“专此答复”关键字，所有是否含有以上关键词决定了是否具体回答了问题。
- 完整性：通过探索数据发现存在一些回复中断的情况，大多数是答复长度短中断在 “回复如下”的关键词这里所以这两个特征可以作为完整性的评价指标。
- 及时性：根据回复时间与留言时间的差值可以得到是否在 15 天内留言，从而判断回复是否及时。

文本特征是指蕴含在文本中、难以直观表现出来、也不能简单统计得出的特征，本文将通过计算留言与答复的主题相似度来评价答复的相关性。高质量的答复与对应的留言应该属于同一个话题，存在较强的关联度。

7.1.1 结构化特征的评分

单项评分：

$$S_i = \frac{Sum_i}{Sum} \times 100$$

（其中 i 代表第几项评分，S_i 代表 i 项评分的分数，Sum_i 表示满足这一项指标的留言数，Sum 表示留言总数）

平均得分：

$$AvgS = \frac{Sum(S_i)}{Max(i)}$$

（其中 i 代表第几项评分，$AvgS$ 代表各项平均得分，$Sum(S_i)$ 表示每项得分之和，$Max(i)$ 表示总共有几项评分）

7.1.2 文本特征的评分

- 7.1.2.1 文本向量表示与 TF-IDF 权重策略

见 5.1.1.3

- 7.1.2.2 LSI (Latent Semantic Indexing)向量

奇异值分解(SVD)：对于一个 m×n 的矩阵 A，可以分解为下面三个矩阵：

$$A_{m\times n} = U_{m\times m} \sum_{k\times k} V^T_{n\times n}$$

有时为了降低矩阵的维度到 k，SVD 的分解可以近似的写为：

$$A_{m\times n} \approx U_{m\times k} \sum_{k\times k} V^T_{k\times n}$$

把上式用到我们的潜在语义索引(LSI)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_{il} 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；$\sum_{lm}$ 对应第 l 个主题和第 m 个词义的相关度。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。

- 7.1.2.3 相似度的计算（余弦相似度）

假定 A 和 B 是两个 n 维向量，A 是 [A1, A2, ..., An] ，B 是 [B1, B2, ..., Bn] ，则 A 与 B 的夹角 θ 的余弦等于：

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

余弦值越接近 1，就表明夹角越接近 0 度，也就是两个向量越相似，文本的相关性就越高。

7.1.3 最终评分

最终的答复质量评分：

$$FS = AvgS \times 0.5 + Similarity \times 100 \times 0.5$$

7.2 模型的求解

7.2.1 求解过程

step1.数据探索：初步探索数据，得到回复长度，格式等信息

step2.初步评价：根据答复长度，答复是否含有”答复如下”等关键词与回答时间判断是否直接、具体、及时的答复了留言

step3.数据加工：对数据进行选取、格式化、jieba 分词等预处理

step4.数据向量化：形成 corpus，并转化成 tf-idf 向量

step5.lsi 模型：利用 lsi 模型将数据转化成 lsi 向量，计算 lsi 值

step6.语义相似度：利用 similarity 模块计算相似度

step7.保存实验结果：整理合并实验结果，并按要求保存

7.2.2 求解结果

最终实现的答复评价表，部分如下所示：

表 11 答复评价表

留言编号	留言主题	留言详情	答复意见	回复相关性	是否直接回复	是否具体回复	是否15天内回复
2549	A2区景蓉华苑物业管理有问题	2019年4月以来，位于A市A2区桂花坪街道的A2区公安分局宿舍区….	现将网友在平台《问政西地省》栏目向胡华衡书记留言反映“A2区景蓉花苑物业…	0.88	是	是	是
2557	在A市买公寓能享受人才新政购房补贴吗？	尊敬的书记：您好！我研究生毕业后根据人才新政落户A市，想买套…	网“A000110735”：您好!您在平台《问政西地省》上的留言已收悉，市住建局及时将您反映…	0.92	是	是	是
2574	关于A市公交站点名称变更的建议	建议将“白竹坡路口”更名为“马坡岭小学”，原“马坡岭小学”取消，保留…	网友“A0009233”，您好，您的留言已收悉，现将具体内容答复如下：关于来信人建议“白竹…	0.93	是	是	否
……							

最终各项得分，如下图所示：

```
直接回复有2339条，占比：0.83，得分：83.06
具体回复有1705条，占比：0.61，得分：60.55
15天内回复有1755条，占比：0.62，得分：62.32
完整的回复有2777条，占比：0.99，得分：98.62
相关度大于0.6的有1933条，占比：0.69，得分：68.64

结构化特征评分：76.14,文本特征评分：68.64
最终得分：72.39
```

图 10 答案评分结果

7.3 结果分析

1.我们可以从表 11 中看到答复的评价结果，可以看到四个评价指标：回复相关性、是否直接回复、是否具体回复、是否 15 天内回复，根据统计其中总共有 2816 条留言，直接回复的有 2339 条，占总留言的 83%，具体回复的有 1705 条，占总留言的 61%，在 15 天内回复的有 1755 条，占总留言的 62%，完整回复的有 2777 条，占总留言的 99%，相关性在 0.6 以上的有 1933 条占 69%。

可以从图 10 中看到结构化特征的平均分 76，相关性（文本特征）得分为 69 分。最终得分为两者的平均分 72 分。

2.在实际比对数据后可以看出因为留言和答复方式的不同以及答复中会含有解释问题的专业语句，所以会产生一些误差，使真实相关性很高的答复计算出来的主题相似度却偏低，这个问题可以在之后的模型改进中继续深入挖掘。

八、模型的评价

8.1 模型的优点

1.模型多次运用比较法得到最优结果。传统学习与深度学习,编辑距离、莱文斯坦比、jaro 距离等多种距离算法,通过多项比较可以得到最适合实验数据的模型，使结果更佳,更有说服力，也增强了模型的可移植性。

2.模型准确性高。任务一中的深度学习准确率达到 0.93，回归模型也达到 0.9 以上。任务二中问题分类的准确性也只出现 1 条错误记录，特定地点/人群识别的效果也很好，在模型的实际应用中准确性才是模型的核心。

3.模型逻辑清晰，可解释性强。每个任务的解题思路清晰，从开始的数据加工到最后结果输出，每一步都有数学原理支撑，简单易懂并有理有据。

4.模型操作方便，使用简单。模型的结构性强，每一步都有详细注释，实际应用时只需更换数据，就可直接使用，具有很好的推广性。

5.模型的可移植性强。模型中应用了自然语言处理的各种主流模块，不仅可以运用在本次场景中，可以快捷方便地移植到其他场景的应用中。

6.传统向量空间模型中只简单地基于单词的出现,但忽视了单词的多义性和同义性,为此本模型采用潜在语义索引(LSI)主题模型来计算文本的主题相似度,很好的避免了这个问题，提高了评估效果。

8.2 模型的缺点

1.模型的数据处理效果还有待提高。实验过程中我们目前能想到的数据处理方式只是单独对留言主题进行处理或者是对留言主题与详情进行拼接；同时，为了简化处理过程，没有考虑“留言主题”和“留言详情”等一些数据的权重。

2.模型中一些模块的运行效率不够高。其中使用的 CNN 与潜在语义索引（LSI）

模型时虽然处理准确性高，但非常耗时。但其他模块的准确率不理想，为了提高实验的准确率，所以本模型还是保留的这两个模块。

九、模型的改进与推广

9.1 模型的改进

1. 继续挖掘新的评价指标，如热度考虑最近时间内，答复质量考虑是否解决了问题;也可以深入考虑模型的各项指标的比重,如频数与点赞反对数的比重、结构化特征与文本特征的比重，可以在改进中赋予不同的比重进行权重调优。

2.在任务三中使用的语义相似度计算模型对提高信息查全率和查准率至关重要。而 LSI 语义相似度计算模型存在不足,所以，可以使用了一种改进的语义相似度计算模型 PHSS[9]。它综合考虑了概念间属性约束和层次结构的差别, 从而更精确地描述了概念之间的语义距离,进而使结果更好。

9.2 模型的推广

1. 本文通过卷积神经网络模型对实验数据进行训练预测，对比正确分类其产生的误差在可接受范围之内，验证了该模型对文本分类归纳具有较好的适用性，同时也可将此模型应用在用户评论、新闻分类等各个分类归纳问题中；同时卷积神经网络不仅适用于文本分类情况，还可用于图像处理，人体行为识别，环视车位检测等研究。

2. 逻辑回归模型是一个被 logistics 方程归一化后的线性回归，预测较为准确，且求出的模型系数易于理解，便于解释；训练速度快，对于稀疏高维特征处理能力较强，可用于新闻个性化推荐系统，CTR 预估等。

3. 命名实体识别模型 FoolNLTK 不仅能够准确提取热点问题相关人物、地区等信息，以便更好地对热点问题进行处理，而且还能将此模型应用在对文章中人物、地区等信息的提取，如《红楼梦》、《三体》等，可以统计相关人物、相关家族、相关企业和相关组织在文中出现的次数占比。

4.本模型运用了主题相似度算法，计算了两个文本的主题相似度，这个思想还可以用于识别文档主题，进行相似文档推荐，文档分类，某类文档搜索等实际应用。

十、参考文献

[1]赵志靖.编辑距离在语言分类研究中的应用[J].现代语文,2018,(05):119-124+2.

[2]刘晓鹏,杨嘉佳,卢凯,田昌海,唐球.面向短文本分类的特征提取与算法研究[J].信息技术与网络安全,2019,38(05):48-52.DOI:10.19358/j.issn.2096-5133.2019.05.010

[3]王宏,门博,雷娜.K 近邻算法在政府采购数据挖掘中的研究与应用[J].智能计算机与应用,2019,9(03):269-272.

[4]曾凡锋,李玉珂,肖珂.基于卷积神经网络的语句级新闻分类算法[J].计算机工程与设计,2020,41(04):978-982.DOI:10.16208/j.issn1000-7024.2020.04.013

[5]刘福刚.一种适用于中文博客自动分类的贝叶斯算法[J].长春师范大学学报,2019,38(12):36-43.

[6]祁小军,兰海翔,卢涵宇,丁蕾锭,薛安琪.贝叶斯、KNN 和 SVM 算法在新闻文本分类中的对比研究[J].电脑知识与技术,2019,15(25):220-222.DOI:10.14004/j.cnki.ckt.2019.3117

[7] 刘宇鹏,栗冬冬.基于 BLSTM-CNN-CRF 的中文命名实体识别方法[J/OL].哈尔滨理工大学学报：115-120https://doi.org/10.15938/j.jhust.2020.01.017.DOI:10.15938/j.jhust.2020.01.0170

[8] 曾小芹.基于 Python 的中文结巴分词技术实现[J].信息与电脑(理论版),2019,31(18):38-39+42.

[9] 刘春辰,刘大有,王生生,赵静滨,王兆丹.改进的语义相似度计算模型及应用[J].吉林大学学报(工学版),2009,39(01):119-123.DOI:10.13229/j.cnki.jdxbgxb2009.01.039

十一、附录

卡方分布的临界x^2值表

自由度	一尾概率（a）									
	0.99	0.98	0.95	0.90	0.50	0.10	0.05	0.03	0.01	0.005
1				0.02	0.45	2.71	3.84	5.02	6.63	7.88
2	0.02	0.05	0.10	0.21	1.39	4.61	5.99	7.38	9.21	10.60
3	0.11	0.22	0.35	0.58	2.37	6.25	7.81	9.35	11.84	12.84
4	0.30	0.48	0.71	1.06	3.36	7.78	9.49	11.14	13.28	14.86
5	0.55	0.83	1.15	1.16	4.35	9.24	11.07	12.83	15.09	16.75
6	0.87	1.24	1.64	2.20	5.35	10.64	12.50	14.45	16.81	18.55
7	1.24	1.69	2.17	2.83	6.35	12.02	14.07	16.01	18.48	20.28
8	1.65	2.18	2.73	3.49	7.34	13.36	15.51	17.53	20.09	21.95
9	2.09	2.70	3.33	4.17	8.34	14.68	16.92	19.02	21.96	23.59
10	2.56	3.25	1.94	4.87	9.34	15.99	18.31	20.48	23.21	25.19
11	3.05	3.82	4.57	5.58	10.34	17.28	19.68	21.92	24.72	26.76
12	3.57	4.40	5.23	6.30	11.34	18.55	21.03	23.34	26.22	28.30
13	4.11	5.01	5.89	7.04	12.34	19.81	22.36	24.74	27.69	29.82
14	4.66	5.63	6.57	7.79	13.34	21.06	23.68	26.12	29.41	31.32
15	5.23	6.27	7.26	8.55	14.34	22.31	25.00	27.49	30.58	32.80
16	5.81	6.91	7.96	9.31	15.34	23.54	26.30	28.85	32.00	34.27
17	6.41	7.56	8.67	10.09	16.34	24.77	27.59	30.19	33.41	35.72
18	7.01	8.23	9.39	10.86	17.34	25.99	28.87	31.53	34.81	37.16
19	7.63	8.91	10.12	11.65	18.34	27.20	30.14	32.85	36.19	38.58
20	8.26	9.59	10.85	12.44	19.34	28.41	31.41	34.17	37.57	40.00
21	8.90	10.28	11.59	13.24	20.34	29.62	32.67	35.48	88.93	41.40
22	9.54	10.98	12.34	14.04	21.34	30.81	33.92	36.78	40.29	42.80
23	10.20	11.69	13.09	14.85	22.34	32.01	35.17	38.08	41.64	44.18
24	10.86	12.40	13.58	15.66	23.34	33.20	36.42	39.36	42.98	15.56
25	11.52	13.12	14.61	16.47	24.34	34.38	37.05	40.65	44.31	46.93
26	12.20	12.84	15.38	17.29	25.34	35.56	38.89	41.92	45.61	48.29
27	12.88	14.57	16.15	18.11	26.34	36.74	40.11	43.19	46.96	49.64
28	13.56	15.31	19.93	18.94	27.34	37.92	41.34	44.46	48.28	50.99
29	14.26	16.05	17.71	19.77	28.34	39.09	42.56	45.72	49.59	52.34
30	14.95	16.79	18.49	20.60	29.34	40.26	43.77	46.98	50.89	53.67

图 12

代码：

任务一：

```
from warnings import filterwarnings
filterwarnings('ignore')  # 忽略警告
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties#解决图表不显示中文
import re
import jieba as jb#导入结巴库
from wordcloud import WordCloud #用于词频云
from sklearn.feature_extraction.text import CountVectorizer as CVec#文本特征提取模块中转化词频向量函数
from sklearn.feature_selection import chi2
```

```
import numpy as np
from sklearn.feature_extraction.text import TfidfTransformer as TTf#文本特征提取模块中转化 tf-idf 权重向量函数
from sklearn.model_selection import train_test_split as tts#用于数据切片
from sklearn.feature_selection import SelectKBest#用于卡方验证选取特征

from sklearn.linear_model import LogisticRegression  # 逻辑回归
from sklearn.neighbors import KNeighborsClassifier  # K 最近邻
from sklearn.svm import SVC  # 支持向量机
from sklearn.tree import DecisionTreeClassifier  # 决策树
from sklearn.ensemble import RandomForestClassifier  # 随机森林
from sklearn.naive_bayes import GaussianNB,MultinomialNB  # 朴素贝叶斯
from sklearn.model_selection import GridSearchCV#网格参数优化
from sklearn.externals import joblib

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
from IPython.display import display
#1.数据探索与预处理
#数据加载
data=pd.read_excel('./C 题全部数据/附件 2.xlsx')
#空值处理
data.isnull().sum()
#去重、选取列
data0=data['留言主题'].drop_duplicates()#去重操作
data_del=data.loc[data0.index,['留言主题','留言详情','一级标签']]
#查看类别的分布,便于之后选取数据
data_lab=data_del['一级标签']#分类标签
lab_sum=data_lab.value_counts()#类别统计
print(data_lab)
print(lab_sum)
# import matplotlib.pyplot as plt
# from matplotlib.font_manager import FontProperties#解决图表不显示中文
font = FontProperties(fname=r"C:/Windows/Fonts/simhei.ttf") #使用 windows 自带黑体字体
lab_sum.plot(x='一级分类', y='count', kind='bar', legend=False,  figsize=(8, 5))
```

```
plt.title("类目数量分布",fontproperties=font)
plt.ylabel('数量', fontsize=16,fontproperties=font)
plt.xlabel('类目', fontsize=16,fontproperties=font)
plt.xticks(fontproperties=font)
plt.show()
#类别数值化
data_del['lab_id']=''#添加存放数值化分类的列
labNUM={}#类别数值对应表，如'城乡建设'：2
i=1
for lab in lab_sum.index:
    labNUM[lab]=i
    i+=1
labNUM2={}#类别数值反向对应表，如 2：'城乡建设'
k=1
for lab in labNUM.keys():
    labNUM2[k]=lab
    k+=1
for i in data_lab.index:
    for j in labNUM.keys():
        if data_lab[i]==j:
            data_del['lab_id'][i]=labNUM[j]
labels=data_del['lab_id']
labels
#平均选取每类数据
n=500
data_1=data_del[data['一级标签']=='城乡建设'].sample(n)#取出 n 条
data_2=data_del[data['一级标签']=='劳动和社会保障'].sample(n)#取出 n 条
data_3=data_del[data['一级标签']=='教育文体'].sample(n)#取出 n 条
data_4=data_del[data['一级标签']=='商贸旅游'].sample(n)#取出 n 条
data_5=data_del[data['一级标签']=='环境保护'].sample(n)#取出 n 条
data_6=data_del[data['一级标签']=='卫生计生'].sample(n)#取出 n 条
data_7=data_del[data['一级标签']=='交通运输'].sample(n)#取出 n 条
data_n=data0=pd.concat([data_1,data_2,data_3,data_4,data_5,data_6,data_7],axis=0)#
拼接数据
data_n
#选出变量与因变量
data_x = data_n['留言主题']+data_n['留言详情']
```

```
print(data_x)
labels=data_n['lab_id']
labels
#无用字符处理
# import re
r1 = '[a-zA-Z0-9’!"#$%&\'()*+,-./:;<=>?@，。?★、…【】《》？\n“”\t‘’！[\\]^_`{|}~]+'
data_x=data_x.apply(lambda x: re.sub(r1, '',x))#去除中文以外字符
data_x
#2.数据加工
#分词处理
# import jieba as jb#导入结巴库
#jb.load_userdict('./newdic1.txt')#向结巴库添加新分词
data_cut=data_x.apply(lambda x: jb.lcut(x))#利用结巴库分词
data_cut
#停用词处理
stopwords=pd.read_csv('./stopwords.txt',header=None,sep='hahaha',encoding='utf-8')#加载停用词表
stopwords=list(stopwords[0])+[' ','　','\xa0','年','月','日','您好','感谢','请','想','市','县','区']#形成最后停用词表
data_stop1=data_cut.apply(lambda x: [i for i in x if i not in stopwords])#去掉数据的停用词
data_stop2=data_stop1.apply(lambda x : ' '.join(x))#将分隔符，转为‘ ’，便于之后数值化处理
data_stop1
#3.数据可视化
#词频统计
wordcount={}#存放词频
for type in labNUM.keys():#分类存放
    wordcount[type]={}
for type in labNUM.keys():#统计每类词频
    for i in data_stop1[data_lab==type]:
        for j in i:
            if j not in wordcount[type]:
                wordcount[type][j]=1
            else:
                wordcount[type][j]+=1
for type in labNUM.keys():#查看每类的词数
    print(type,':',len(wordcount[type]))
    #画词云图
```

```
# from wordcloud import WordCloud #用于词频云
# from matplotlib.font_manager import FontProperties#解决图表不显示中文
font = FontProperties(fname=r"C:/Windows/Fonts/simhei.ttf") #使用 windows 自带黑体字体

bg=plt.imread('./bg.jpg')#加载背景图
wc=WordCloud(mask=bg,background_color='white',font_path='C:/Windows/Fonts/simhei.ttf')#设置背景图，颜色，字体；字体必须设
n=1    # add_subplot() 不能从 0 开始
plt.figure(dpi=300) #通过这里可以放大或缩小
for type in labNUM.keys():#按类别画图
    wc.fit_words(wordcount[type])
    plt.subplot(3,3,n)
    plt.title(type,fontproperties=font)
    plt.imshow(wc)#绘图
    plt.axis('off')
    n+=1
#卡方验证
# from sklearn.feature_extraction.text import CountVectorizer as CVec#文本特征提取模块中转化词频向量函数
cvec=CVec()
features=cvec.fit_transform(data_stop2)#调用对应方法将训练集转化为训练词频向量

# from sklearn.feature_selection import chi2
# import numpy as np
N = 100
for lab_id in labNUM.values():
    features_chi2 = chi2(features,labels == lab_id)
    indices = np.argsort(features_chi2[0])
    feature_names = np.array(cvec.get_feature_names())[indices]
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
    print("_____________________________________________________________")
    print("【",labNUM2[lab_id],"】",'这一类中：')
    print("最相关的{}词条:\n{}".format(N,unigrams[-N:]))
#4.数据向量化
# from sklearn.feature_extraction.text import TfidfTransformer as TTf#文本特征提取模块中转化 tf-idf 权重向量函数
# from sklearn.model_selection import train_test_split as tts#用于数据切片
```

```
traindata,testdata,lab_tr,lab_te=tts(data_stop2,labels,test_size=0.2)#将数据分为训练集与测试集,测试集占五分之一
cvec1=CVec()#先把模型初保存，因为测试集与训练集的列数（维度）不一样（切片随机所以不一样），之后测试集要用这个转换

traindata=cvec1.fit_transform(traindata)#调用对应方法将训练集转化为训练词频向量
x_tr=TTf().fit_transform(traindata.toarray()).toarray()#调用对应方法转化为 tf-idf 权重训练词频向量

testdata=CVec(vocabulary=cvec1.vocabulary_).fit_transform(testdata)#调用对应方法将测试集转化为测试词频向量
x_te=TTf().fit_transform(testdata.toarray()).toarray()#调用对应方法转化为 tf-idf 权重训练词频向量

y_tr=list(lab_tr)#将标签转成列表，因为模型训练不接受 Series 型数据
y_te=list(lab_te)
#5.卡方验证进行特征选取(初设 2500)
# from sklearn.feature_selection import SelectKBest
ch2 = SelectKBest(chi2, k=2500)
x_tr_ch2 = ch2.fit_transform(x_tr, y_tr)
x_te_ch2 = ch2.transform(x_te)
#6.模型选择
# from sklearn.linear_model import LogisticRegression  # 逻辑回归
# from sklearn.neighbors import KNeighborsClassifier  # K 最近邻
# from sklearn.svm import SVC  # 支持向量机
# from sklearn.tree import DecisionTreeClassifier  # 决策树
# from sklearn.ensemble import RandomForestClassifier  # 随机森林
# from sklearn.naive_bayes import GaussianNB,MultinomialNB  # 朴素贝叶斯

# 建模、设定参数
classifiers = [
    ('Logistic Regression', LogisticRegression(class_weight='balanced')),  # 逻辑回归
    ('Nearest Neighbors', KNeighborsClassifier(n_neighbors=4,metric='minkowski')),  # K 最近邻
    ('SVM', SVC(class_weight='balanced')),  # 支持向量机
    ('Decision Tree', DecisionTreeClassifier(max_depth=10)),  # 决策树
    ('Random Forest', RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1)),  # 随机森林
```

```
    ('Gaussian NB', GaussianNB()),  # 高斯贝叶斯
    ('MultinomialNB NB', MultinomialNB(alpha=1)),  # 朴素贝叶斯
]
score={}
 # 遍历分类器
for name, clf in classifiers:
    clf.fit(x_tr_ch2,y_tr)  # 训练
    s = clf.score(x_te_ch2,y_te)  # 模型评分
    score[name]=s
score
#7.模型调优
lr = LogisticRegression()
# from sklearn.model_selection import GridSearchCV#网格参数优化
#调参数 C,max_iter(liblinear 情况下)
print("==========下面是 GridSearchCV 的测试结果==============")
C= [0.1, 1, 10, 100]
class_weight=['balanced']
max_iter=[1,10,20]
param_grid=dict(C=C, class_weight=class_weight,max_iter=max_iter)
grid_search=GridSearchCV(lr,param_grid=param_grid)
grid_search.fit(x_tr_ch2,y_tr)
best_lr0=grid_search.best_estimator_
lr0_score=grid_search.score(x_te_ch2,y_te)
# 预测准确率
print('测试集上的准确率:',lr0_score)
print('在交叉验证中最好的结果:',grid_search.best_score_)
print('选择最好的模型是:',best_lr0)
#调参数 solver
print("==========下面是 GridSearchCV 的测试结果==============")
C= [best_lr0.C]
class_weight=['balanced']
solver=['newton-cg', 'lbfgs', 'sag']
max_iter=[best_lr0.max_iter]
multi_class=['multinomial']
param_grid=dict(C=C,
class_weight=class_weight,solver=solver,max_iter=max_iter,multi_class=multi_class)
grid_search=GridSearchCV(lr,param_grid=param_grid)
```

```
grid_search.fit(x_tr_ch2,y_tr)
best_lr1=grid_search.best_estimator_
lr1_score=grid_search.score(x_te_ch2,y_te)
# 预测准确率
print('测试集上的准确率:',lr1_score)
print('在交叉验证中最好的结果:',grid_search.best_score_)
print('选择最好的模型是:',best_lr1)
if lr0_score>=lr1_score:
    best_lr=best_lr0
else:
    best_lr=best_lr1
#特征调优
k=[2500,10000,15000,20000]
best_k=0
best_score=0
for i in k:
    ch2 = SelectKBest(chi2, k=i)
    x_tr_ch2 = ch2.fit_transform(x_tr, y_tr)
    x_te_ch2 = ch2.transform(x_te)
    best_lr.fit(x_tr_ch2,y_tr)   # 训练
    score=best_lr.score(x_te_ch2,y_te)
    print(i,':',score)   # 模型评分
    if score>=best_score:
        best_k=i
        best_score=score
best_k
ch2 = SelectKBest(chi2, k=best_k)
x_tr_ch2 = ch2.fit_transform(x_tr, y_tr)
x_te_ch2 = ch2.transform(x_te)
best_lr.fit(x_tr_ch2,y_tr)
#7.模型保存
# from sklearn.externals import joblib
joblib.dump(best_lr,'firstTDbestModel.pkl')
#8.模型预测与评估
#使用测试集预测结果
y_pred = best_lr.predict(x_te_ch2)
y_pred
```

```
#模型评估
# from sklearn.metrics import accuracy_score, confusion_matrix
# from sklearn.metrics import classification_report
# import seaborn as sns

#生成混淆矩阵
conf_mat = confusion_matrix(y_te, y_pred)
plt.subplots(figsize=(10,8))
sns.heatmap(conf_mat, annot=True, fmt='d',
                xticklabels=labNUM.keys(), yticklabels=labNUM.keys())
plt.ylabel('实际结果',fontsize=18,fontproperties=font)
plt.xlabel('预测结果',fontsize=18,fontproperties=font)
plt.xticks(fontproperties=font)
plt.yticks(fontproperties=font)
plt.show()

#准确率，召回率，f1_score
print(classification_report(y_te, y_pred,target_names=labNUM.keys()))
#9.查看错误预测的具体记录
# from IPython.display import display
for actual in labNUM2.keys():
    for predicted in   labNUM2.keys():
        if (predicted != actual) and (conf_mat[actual-1, predicted-1]>0) :

print("##########################################################################
##################################################")
                print("【{}】预测为【{}】: {} 例.".format(labNUM2[actual],labNUM2[predicted],
conf_mat[actual-1, predicted-1]))
                x=0
                err_index=[]
                for lab in lab_te.index:
                    if (lab_te[lab]==actual and y_pred[x]==predicted and lab_te[lab] !=
y_pred[x]):
                        err_index.append(lab)
                    x=x+1
                display(data_stop2.loc[err_index])
                display(data_del.loc[err_index])
```

任务二：

```
import Levenshtein#计算相似度
from warnings import filterwarnings
filterwarnings('ignore')  # 忽略警告
import pandas as pd
import fool
import re
import jieba as jb#导入结巴库
import datetime
#1.加载数据
# import pandas as pd
data_save=pd.read_excel('./C 题全部数据/附件 3.xlsx')
data_save.head()
#数据准备
data_save.insert(0, column = "问题 ID", value = None)
for i in data_save.index:#根据留言主题的长度排序,为之后相似度比较做准备
    data_save['问题 ID'][i]=len(data_save['留言主题'][i])
data_save.sort_values('问题 ID', inplace=True)
data_save['问题 ID']= None#将问题 ID 清除
data_save.head()
data0=data_save['留言主题']
dataindex=[]#将索引保存到列表供之后更新使用
for i in data0.index:
    dataindex.append(i)
#3.根据相似度分类
# import re
r1 = '[a-zA-Z0-9’!"#$%（）·&\'()*+,-./:;<=>?@，。?★、…【】《》？ \n“”\t‘’！ [\\]^_`{|}~]+'
data_x=data0.apply(lambda x: re.sub(r1, '',x))#去除中文以外字符
data_cut=data_x.apply(lambda x: jb.lcut(x))#利用结巴库分词
wordcount={}#存放词频
for i in data_cut:
    for j in i:
        if j not in wordcount:
            wordcount[j]=1
        else:
            wordcount[j]+=1
sorted(wordcount.items(),key=lambda item:item[1],reverse=True)
```

```
#根据词频人工选停用词
stopwords=['市','的','区','小区','在','问题','县','了','投诉','解决','反映','咨询','建议','关于','请','吗','能','还','请求','请问','希望']
data_stop=data_cut.apply(lambda x: [i for i in x if i not in stopwords])#去掉数据的停用词
data_stop=data_stop.apply(lambda x : ''.join(x))#将数据转为字符串，便于之后相似度比较
data_stop
#将每一条数据与之后的数据进行两两相似度比较
# import Levenshtein#计算相似度
id=10#问题种类
for str1 in dataindex:
    list=[]#存放同一种问题的索引
    for str2 in dataindex:
        sim = Levenshtein.jaro(data_stop[str1],data_stop[str2])#比较两条问题相似度
        if sim>0.545:#相似度大于 0.545 为同一种问题
            list.append(str2)
    for i in list:
        dataindex.remove(i)#更新数据，将这一类问题从之后的数据中删除
    data_save.loc[list,'问题 ID']=id
    id+=1
data_save.head()
#4.统计热点问题
data_save.dropna(subset=['问题 ID'],inplace=True)#将没有归类的问题删除
hot=data_save['问题 ID'].value_counts()#每类问题的频数
#统计频数与点赞、反对数之和
for i in hot.index:
    for j in data_save.index:
        if data_save.loc[j,'问题 ID']==i:
            hot[i]=hot[i]+data_save.loc[j,'点赞数']+data_save.loc[j,'反对数']
hot.sort_values(ascending = False,inplace = True)#根据计算之和将问题排序
hot=hot.head()#取出热度为前五名的问题
hot
for i in data_save.index:#删除不是前五名的问题
    if(data_save.loc[i,'问题 ID'] not in hot.index):
        data_save.drop(i,inplace = True)
#表 2 输出格式整理
k=1#将问题 id 重新置为 1~5
for i in hot.index:
```

```
    for j in data_save.index:
        if data_save.loc[j,'问题 ID']==i:
            data_save.loc[j,'问题 ID']=k
    k+=1
data_save.sort_values('问题 ID', inplace=True)#根据问题编号排序
data_save.head(50)
#5.保存数据为表 2
data_save.to_excel('热点问题留言明细表.xlsx',index = False)
#6.统计热度
hot=data_save['问题 ID'].value_counts()#更新 hot 表，为了整理输出格式
for i in hot.index:
    for j in data_save.index:
        if data_save.loc[j,'问题 ID']==i:
            hot[i]=hot[i]+data_save.loc[j,'点赞数']+data_save.loc[j,'反对数']
hot.sort_values(ascending = False,inplace = True)#根据计算之和重新排序
hotdetail={}#存放热度指标
times={}#存放每类的所有时间，为下一步准备数据
for i in hot.index:
    hotdetail[i]={'总热度':hot[i],'提出人数':0,'点赞':0,'反对':0}
    times[i]=[]
    for j in data_save.index:
        if data_save.loc[j,'问题 ID']==i:
            str2date=datetime.datetime.strptime(data_save.loc[j,'  留  言  时  间
'],'%Y/%m/%d %H:%M:%S').date()#字符串转化为 date 形式
            times[i].append(str2date)
            hotdetail[i]['点赞']=hotdetail[i]['点赞']+data_save.loc[j,'点赞数']
            hotdetail[i]['反对']=hotdetail[i]['反对']+data_save.loc[j,'反对数']
    hotdetail[i]['提出人数']=hotdetail[i]['总热度']-hotdetail[i]['点赞']-hotdetail[i]['反对']
    hotdetail[i]=str(hotdetail[i])
hotdetail=pd.DataFrame(hotdetail,index=['热度指数']).T
hotdetail
#7.计算时间段
time={}
for i in times.keys():
    time[i]=str(min(times[i]))+'至'+str(max(times[i]))
time=pd.DataFrame(time,index=['时间范围']).T
time
```

```
#8.提取问题描述,将点赞数最多的作为这类问题的描述
question={}
for i in hot.index:
    index=data_save[data_save['问题 ID']==i]['点赞数'].idxmax()
    question[i]=data_save.loc[index,'留言主题']
question=pd.DataFrame(question,index=['问题描述']).T
question
#9.提取地点/人群
# import fool
own = {}#准备地点/人群提取的数据
for i in hot.index:
    own[i]= []
    for j in data_save.index:
        if data_save.loc[j,'问题 ID']==i:
            own[i].append(data_save.loc[j,'留言主题'])
for i in own.keys():
    words, ners = fool.analysis(own[i])
    own[i]=[x for x in ners if x!=[]]
    if(own[i]):
        own[i]=pd.value_counts(own[i]).index[0][0][3]#根据频数排序，取最高频数的名称
    else:
        own[i]='无'
own=pd.DataFrame(own,index=['地点/人群']).T
own
#10.按格式合并数据，形成表 1
hotdetail.insert(0, column = "热度排名", value = hotdetail.index)
hotdetail.insert(1, column = "问题 ID", value = hotdetail.index)
hotdetail['时间范围']=time
hotdetail['地点/人群']=own
hotdetail['问题描述']=question
hotdetail
#5.保存数据为表 1
hotdetail.to_excel('热点问题表.xlsx',index = False)
```

任务三：

```
import pandas as pd
import re
import jieba
```

```
from gensim import corpora, models
from gensim.similarities import Similarity
from gensim import similarities
import datetime
#1.数据加载
data=pd.read_excel('./C 题全部数据/附件 4.xlsx')
data.insert(7,column = "回复相关性", value = None)
data.insert(8,column = "是否直接回复", value = None)
data.insert(9,column = "是否具体回复", value = None)
data.insert(10,column = "是否 15 天内回复", value = None)
data.head()
#时间格式统一
for a in data.index:
    s=data.loc[a,'留言时间']
    f=data.loc[a,'答复时间']
    if(type(s) is not str):
        s=s.strftime('%Y/%m/%d %H:%M:%S')
    if(type(f) is not str):
        f=f.strftime('%Y/%m/%d %H:%M:%S')
    s=s.split()
    f=f.split()
    data.loc[a,'留言时间']=s[0]
    data.loc[a,'答复时间']=f[0]
data.head()
#2.回复内容、时间初步判断
data_a=data['答复意见']
NUM=0
for a in data_a.index:
    s=data.loc[a,'留言时间']
    f=data.loc[a,'答复时间']

time=datetime.datetime.strptime(f,'%Y/%m/%d')-datetime.datetime.strptime(s,'%Y/%m/%
d')
    if time.days>15:
        data.loc[a,'是否 15 天内回复']='否'
    else:
        data.loc[a,'是否 15 天内回复']='是'
```

```
    if(len(data_a[a])>100):
        data.loc[a,'是否直接回复']='是'
        if ('回复如下' in data_a[a]) or ('专此回复' in data_a[a]) or ('专此答复' in data_a[a])
or ('答复如下' in data_a[a]):
            data.loc[a,'是否具体回复']='是'
        else:
            data.loc[a,'是否具体回复']='否'
    else:
        data.loc[a,'是否直接回复']='否'
        data.loc[a,'是否具体回复']='否'
        if ('回复如下' in data_a[a]) or ('专此回复' in data_a[a]) or ('专此答复' in data_a[a])
or ('答复如下' in data_a[a]):
            NUM=NUM+1
data
#3.回复相关性评价
#数据加工
data_x=data['留言主题']+data['留言详情']
r1 = '[a-zA-Z0-9’!"#$（）%&\'()*+,-./:;<=>?@，。?★、…【】《》？ \n“”\t‘’！ [\\]^_`{|}~]+'
data_x=data_x.apply(lambda x: re.sub(r1, '',x))#去除中文以外字符
data_cut=data_x.apply(lambda x: jieba.lcut(x))#利用结巴库分词
stopwords=pd.read_csv('./stopwords.txt',header=None,sep='hahaha',encoding='utf-8')#加
载停用词表
stopwords=list(stopwords[0])+[' ','　','\xa0','年','乡','月','日','您好','感谢','请','想','市','县','区','
市区','\u3000']#形成最后停用词表
data_stop=data_cut.apply(lambda x: [i for i in x if i not in stopwords])#去掉数据的停用词
for i in data_stop.index:#添加 key 值,便于之后形成字典
    data_stop[i].insert(0,str(i))
data_stop
# 生成字典和向量语料
corpora_documents = []
for item in data_stop:
    corpora_documents.append(item)
dictionary = corpora.Dictionary(corpora_documents)#形成字典
# 得到语料中每一条记录对应的稀疏向量（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
# corpus 是一个返回 bow 向量的迭代器。
corpus = [dictionary.doc2bow(text) for text in corpora_documents]#形成词袋
```

```
#转化成 tf-idf 向量，下面代码将完成对 corpus 中出现的每一个特征的 IDF 值的统计工作
tfidf_model=models.TfidfModel(corpus)#转为 Tfi 向量
corpus_tfidf = [tfidf_model[doc] for doc in corpus]
#建立 lsi 模型
lsi= models.LsiModel(corpus_tfidf,id2word=dictionary,num_topics=50)
#同样预处理测试数据
test_data=data['答复意见']
test_data=test_data.apply(lambda x: re.sub(r1, '',x))
test_cut=test_data.apply(lambda x: jieba.lcut(x))
test_stop=test_cut.apply(lambda x: [i for i in x if i not in stopwords])
#计算语义相似度
simlist={}
for test in test_stop.index:
    test_cut_raw_1 = test_stop[test]
    test_corpus_3 = dictionary.doc2bow(test_cut_raw_1)
    test_corpus_tfidf_3 = tfidf_model[test_corpus_3]#同样转为 tf-idf 向量
    #转化成 lsi 向量，计算 lsi 值
    test_corpus_lsi_3 = lsi[test_corpus_tfidf_3]
    #相似度查询
    index = similarities.MatrixSimilarity(lsi[corpus]) #计算 lsi 值，保存索引
    sims = sorted(enumerate(index[test_corpus_lsi_3]), key=lambda item: -item[1])#计算
相似度，排序并添加索引
    for s in sims:#找出对应记录的相似值
        if s[0]==test:
            simlist[test]=s[1]
data['回复相关性']=simlist.values()
#4.结果保存
data_s=data[['留言编号','留言主题','留言详情','答复意见','回复相关性','是否直接回复','是否
具体回复','是否 15 天内回复']
data_s
#5.计算得分
sum=data_s.shape[0]
n=data_s[data_s['是否直接回复']=='是'].shape[0]
s1=n/sum
print('直接回复有%d 条，占比：%.2f，得分：%.2f' %(n,s1,s1*100))
n=data_s[data_s['是否具体回复']=='是'].shape[0]
s2=n/sum
```

```
print('具体回复有%d 条，占比：%.2f，得分：%.2f' %(n,s2,s2*100))
n=data_s[data_s['是否 15 天内回复']=='是'].shape[0]
s3=n/sum
print('15 天内回复有%d 条，占比：%.2f，得分：%.2f' %(n,s3,s3*100))
s4=(sum-NUM)/sum
print('完整的回复有%d 条，占比：%.2f，得分：%.2f' %(sum-NUM,s4,s4*100))
n=data_s[data_s['回复相关性']>=0.6].shape[0]
similarity=n/sum
print('相关度大于 0.6 的有%d 条，占比：%.2f，得分：%.2f' %(n,similarity,similarity*100))
avgS=(s1+s2+s3+s4)/4*100
finalScore=(similarity*100+avgS)/2
print('结构化特征评分：%.2f,文本特征评分：%.2f' %(avgS,similarity*100))
print('最终得分：%.2f' %(finalScore))
```