“智慧政务”中的文本挖掘应用

摘要

本文旨在设计运用自然语言处理和文本挖掘等方法使计算机能够解决群众留言自动分类，热点问题挖掘，答复意见的质量评价等任务。近年来，随着互联网的广泛应用和网络问政的迅速发展，网络问政平台已经成为政府获取群众信息的主要渠道。因此，运用网络文本分析和数据挖据对群众问政留言记录及相关部门对部分群众留言的答复意见的研究具有重大意义。

针对问题一，首先，将附件 2 中列名为“留言详情”的内容作为特征，列名为“一级分类”的内容作为标签，建立关于留言内容分类的 **BERT-wwm 模型**，得到每个测试数据属于所有类别的概率值矩阵，矩阵中概率值最大的列即为所属类别，使用**查准率、查全率、F_1-score** 对分类结果进行评价，计算出**查准率**的值为 0.92，**查全率**的值为 0.91，**F_1-score** 的值为 0.91。

针对问题二，本文通过建立热度评价模型：**热度=w1*该热点留言总数+w2*（总点赞数-总反对数）/该热点留言总数**，根据附件 3 提供的内容，使用 **DBSCAN 聚类模型**根据留言主题和留言详情摘要对留言内容进行聚类，得到 27 个聚类簇，对每个聚类簇进行排名，得到前 5 个**热点问题**，提取时间范围和地点人群的信息，使用 **TextRank 算法**提取每一簇里面得分最高的留言主题作为该热点的问题描述，然后分类汇总每条留言。

针对问题三，本文根据附件 4 所给数据，对答复的**相关性、完整性、可解释性，及时性**进行定义，建立对答复意见质量的综合评价模型：**质量评分=w1*语义相似度+w2*文本长度（标准化）+w3*引用解释依据的数量（标准化）+w4*答复时间间隔（标准化）**，语义相似度是通过 **Doc2vec 模型**对留言和答复内容进行文本向量化后利用**余弦距离**计算得到的；最后，设定阈值，质量评分大于 0.2 的为高质量评价，小于 0.1 的为低质量评价，介于两者之间则为中等质量评价。

关键词：智慧政务；BERT-wwm 模型；热点问题；DBSCAN 密度聚类；TextRank 算法；Doc2vec 模型；余弦距离。

Abstract

This article aims to design the use of natural language processing and text mining methods to enable the computer to solve tasks such as automatic classification of mass messages, mining of hot issues, and quality evaluation of answers. In recent years, with the widespread use of the Internet and the rapid development of online questioning, online questioning platforms have become the main channel for the government to obtain mass information. Therefore, it is of great significance to use network text analysis and data mining to study the records of the masses 'political messages and the relevant departments' responses to some masses 'messages.

For question one, first, take the content listed in Annex 2 as "message details" as features, and the content listed as "first-class classification" as tags, establish a BERT-wwm model for the classification of message content, and get each test data belonging to all categories The probability value matrix of the matrix, the column with the largest probability value in the matrix is the category. Use the precision rate, recall rate, and F1-score to evaluate the classification results, calculate the precision rate value is 0.92, the recall rate value is 0.91, and the value of F1-score is 0.91.

For question two, In this paper, through the establishment of a heat evaluation model, heat = w1 * total number of hotspot messages + w2 * (total likes-total opposition) / total number of hotspots. According to the content provided in Annex 3, use the DBSCAN clustering model according to the subject of the message and The message details summary clusters the content of the message to get 27 clusters, ranks each cluster, gets the top 5 hotspot questions, extracts the time range and location crowd information, and uses the TextRank algorithm to extract each cluster The subject of the message with the highest score is used as the problem description of the hotspot, and then each message is classified and summarized.

In response to question three, This article defines the relevance, completeness, interpretability, and timeliness of the response based on the data given in Annex 4, and establishes a comprehensive evaluation model for the quality of the response opinion: quality score = w1 * semantic similarity + w2 * text length

((Standardization) + w3 * Number of citation explanations (standardization) + w4 * Reply time interval (standardization), semantic similarity is calculated by the Doc2vec model after text vectorization of the message and reply content using cosine distance calculation; At a certain threshold, a quality score greater than 0.2 is a high-quality evaluation, a value less than 0.1 is a low-quality evaluation, and an intermediate quality evaluation is in between.

Key word: Smart government affairs; BERT-wwm model; DBSCAN clustering; TextRank algorithm; Doc2vec model； cosine distance

目录

1 问题重述

1.1 问题背景

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、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 要解决的问题

1) 根据附件2给出的数据，建立有关于留言内容的一级标签分类模型。通过建立好的模型，对留言内容进行分类，使用F_1-score对模型的分类效果进行评价。

2）根据附件3对留言内容进行聚类，再对聚好类的结果进行热度分析，通常是定义一个热度指数，得到排名前五的热点问题，并给出结果，汇总留言信息，整理成相关表格。

3）根据附件4，对留言的答复意见进行相关性、完整性、可解释性和及时性方面的评价，从而得出对答复意见的质量的评价方案。

2 问题一：群众留言分类

2.1 问题分析

首先，需要提取附件 2 中的留言内容及其对应的标签内容，对数据进行预处理，整理成可用于 bert 模型的数据集，按 8:1:1 的比例进行分层随机抽样将数据集划分为 train(训练集)，test(测试集)，dev(验证集)，利用该数据集建立有关于留言内容的一级标签分类 bert 模型，并对模型效果进行评价。

2.2 模型建立

2.2.1 BERT-wwm 模型

BERT-wwm 模型，即中文全词覆盖的（Whole Word Masking）BERT 预训练模型，它在 BERT 模型的基础上进行了小升级，主要更改了原预训练阶段的训练样本生成策略。BERT(Bidirectional Encoder Representations fromTransformers)模型[1] 是由 Google 于 2018 年提出的，模型采用了表义能力更强的 Transformer 网络结构[2]来对语言模型进行训练，是一种通过大量语言资料训练得来的一个模型，是第一个用于在预训练 NLP 上的无监督的、深度双向系统。

Devlin J 等[1]设计的 BERT 模型主要分为两大部分，主要分为输入层和双向 Transformer 编码层，如下图 2-1 所示：

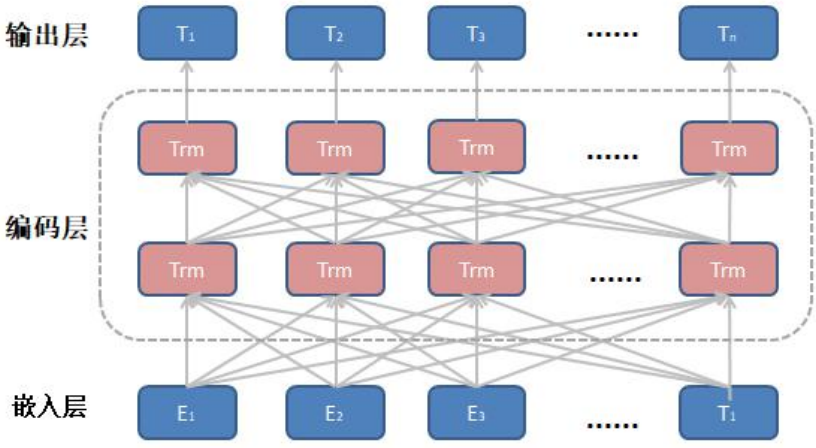

图 2-1 BERT 模型基本结构示意图

2.2.2 Bert 网络框架

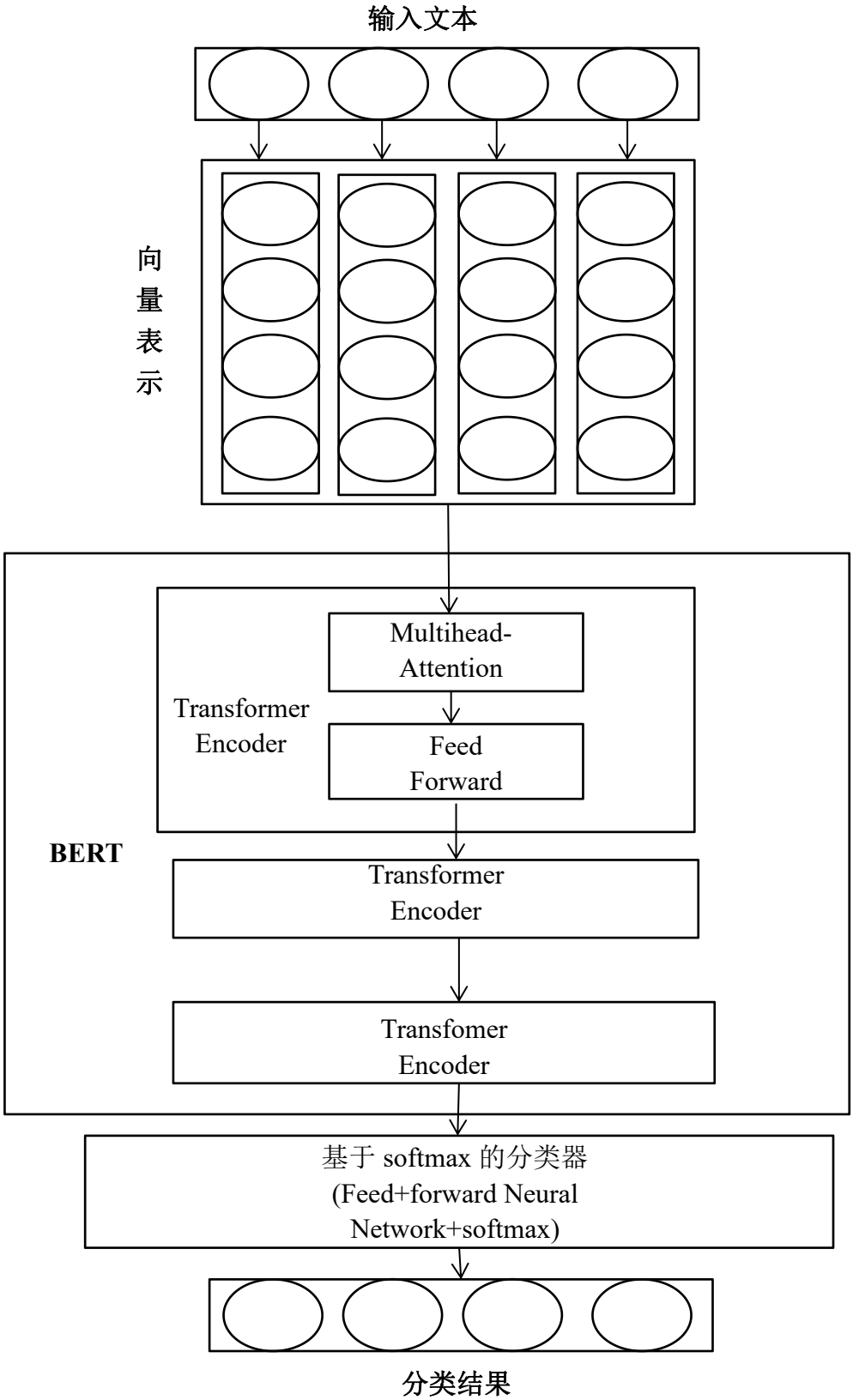

图 2-2　BERT 模型网络框架

2.2.3 Bert 所用 Transformer 内部结构图

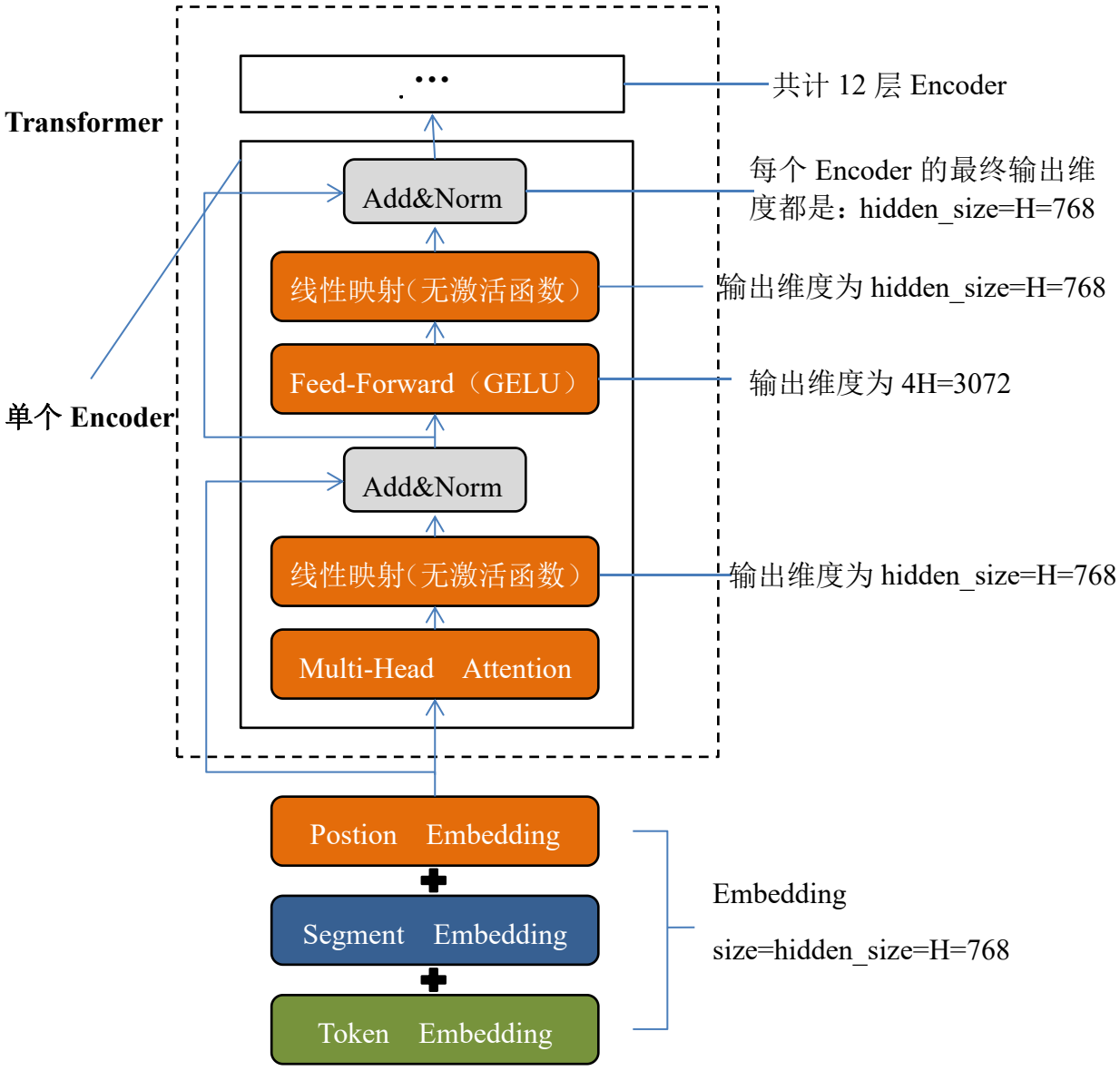

图 2-3 Transformer 内部结构图

由于谷歌官方发布的 BERT-base（Chinese）中，中文是以字为粒度进行切分，没有考虑中文需要分词的特点。应用全词 mask，而非字粒度的中文 BERT 模型可能有更好的表现，因此我们在这里采用，哈工大讯飞联合实验室发布的全词覆盖的中文 BERT 预训练模型(中文 BERT-wwm)。

下面展示了全词 Mask 的生成样例。 注意：为了方便理解，下述例子中只考虑替换成[MASK]标签的情况：

表 1-1 全词 Mask 的生成样例

说明	样例
原始文本	使用语言模型来预测下一个词的 probability。
分词文本	使用 语言 模型 来 预测 下 一个 词 的 probability。
原始 Mask 输入	使 用 语 言 [MASK] 型 来 [MASK] 测 下 一 个 词的 pro [MASK] ##lity。
全词 Mask 输入	使 用 语 言 [MASK] [MASK] 来 [MASK] [MASK] 下 一 个 词的[MASK] [MASK] [MASK]。

本文模型在预训练的 BERT-wwm 模型的基础上进行了结构的微调，使其能更好的应用于本次文本分类任务。

2.3 数据预处理

2.3.1 数据清洗

（1）读取数据并去除文本两端空格：运用 pandas 包中 read_excel 导入附件 2 中的 4，5 列，即留言详情与一级分类，由于留言详情中内容两端存在大量无意义的空格占据大量字符长度，因此对留言详情中的内容运用 str.strip()去除两端空格。

（2）文本去重：经过排查，留言详情中存在重复行，且对应的标签内容不一致，因此对于重复的数据进行删除

（3）对数据进行空值和异常值的排查

2.3.2 数据集划分

对清洗完的数据按照 8：1：1 的比例按标签类别进行分层随机抽样，将数据集划分为训练集(train.tsv)，测试集(test.tsv)，开发集(dev.tsv)。(设置随机种子，保证每次随机的结果都是一样的)

将数据集导出为以下形式：

2.4 模型求解

2.4.1 实验条件

表 2-3 实验环境配置情况

实验工具	型号或版本
显卡	Tesla P100-PCIE-16GB
操作系统	Linux
编程语言	Python 3.6.9
深度学习框架	Tensorflow 1.15.0

2.4.2 数据选择

对于数据预处理后生成的 3 组数据集，train.tsv 作为训练集进行模型训练。dev.tsv 作为训练过程中的验证集，验证模型拟合效果。test.tsv 作为模型训练完成后的测试集，对模型的分类效果进行评价。

2.4.3 算法流程

整体算法流程图

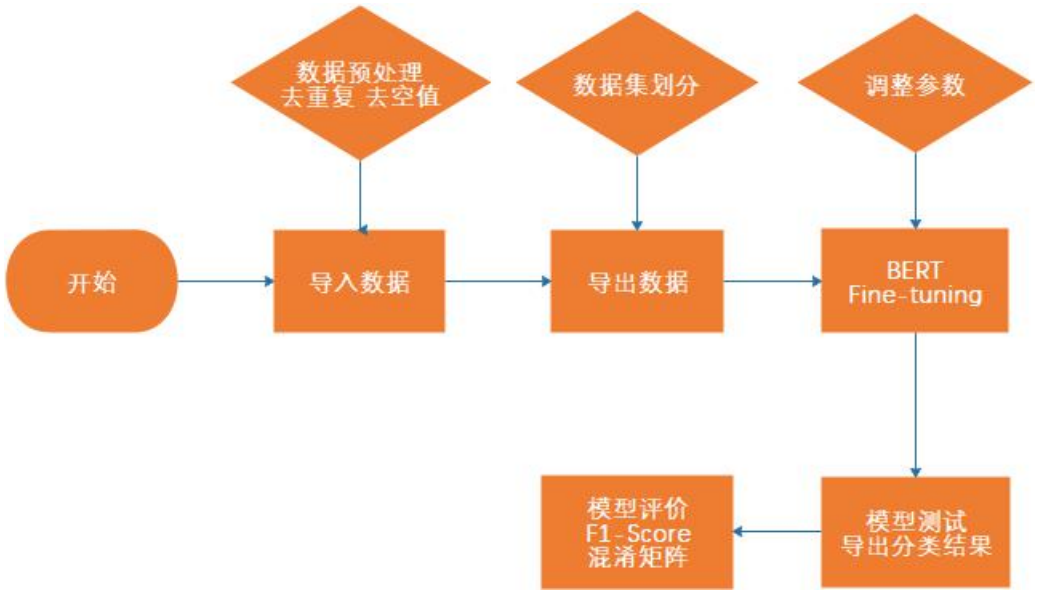

图 2-4 算法流程图

2.4.4 程序框架

BERT 本质上是一个两段式的 NLP 模型。第一个阶段叫做：Pre-training，跟 WordEmbedding 类似，利用现有无标记的语料训练一个语言模型。第二个阶段叫做：Fine-tuning，利用预训练好的语言模型，完成具体的 NLP 下游任务。Google 已经投入了大规模的语料和昂贵的机器帮我们完成了 Pre-training 过程

在 fine-tuning 过程这里使用 run_classifier.py 进行句子分类任务。从主函数开始，可以发现它指定一些必须的参数：“data_dir”，“task_name”，“vocab_file”，“bert_config_file”，“output_data”。

```
if __name__ == "__main__":
    flags.mark_flag_as_required("data_dir")
    flags.mark_flag_as_required("task_name")
    flags.mark_flag_as_required("vocab_file")
    flags.mark_flag_as_required("bert_config_file")
    flags.mark_flag_as_required("output_dir")
    tf.app.run()
```

BERT 代码中 processor 就是负责对模型的输入进行处理。这样，我们需要在在原本 main 函数的 processor 字典里，加入自定义的 processor 类，取名为

MyTaskProcessor，即可在运行参数里指定调用该 processor。

```
def main(_):
    tf.logging.set_verbosity(tf.logging.INFO)
    processors = {
        "cola": ColaProcessor,
        "mnli": MnliProcessor,
        "mrpc": MrpcProcessor,
        "xnli": XnliProcessor,
        "mytask": MyTaskProcessor,#自定义的 Processor
    }
```

自定义的 processor 里需要继承 DataProcessor，并重载获取 label 的 get_labels 和获取单个输入的 get_train_examples，get_dev_examples 和 get_test_examples 函数。其分别会在 main 函数的 FLAGS.do_train 、 FLAGS.do_eval 和 FLAGS.do_predict 阶段被调用。

之后就可以直接运行 run_classsifier.py 进行模型的训练。在运行时需要制定一些参数。

2.4.5 模型的训练

经过分析，由于留言详情字符串长度百分之 75 分位数为: 465.0，为了模型有更好的分类效果，因此 max_seq_length 设置为 bert 模型所能设置的最大值 512，learning_rate 则设置为由许多专业人士测试过的篇章级文本分类最佳学习率 2e-5

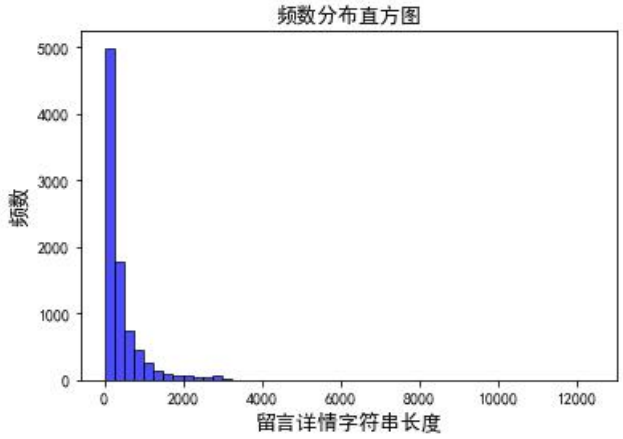

图 2-5 频数分布直方图

表 2-4 模型主要参数

参数	值
task_name	mytask
do_train	true
do_eval	true
Vocab_file	chinese-bert_chinese_wwm_L-12_H-768_A-12/vocab.txt
bert_config_file	chinese-bert_chinese_wwm_L-12_H-768_A-12/bert_config.json
init_checkpoint	chinese-bert_chinese_wwm_L-12_H-768_A-12/bert_model.ckpt
max_seq_length	512
train_batch_size	8
learning_rate	2e-5
run_train_epochs	5
output_dir	output

部分训练过程展示如下图 2-6 所示：

```
I0501 02:35:59.372569 140171572623232 run_classifier.py:926] ***** Running training *****
INFO:tensorflow:  Num examples = 6123
I0501 02:35:59.372847 140171572623232 run_classifier.py:927]   Num examples = 6123
INFO:tensorflow:  Batch size = 8
I0501 02:35:59.373695 140171572623232 run_classifier.py:928]   Batch size = 8
INFO:tensorflow:  Num steps = 15307
I0501 02:35:59.373827 140171572623232 run_classifier.py:929]   Num steps = 15307
```

```
INFO:tensorflow:*** Example ***
I0317 12:00:49.490914 140397564315520 run_classifier.py:515] *** Example ***
INFO:tensorflow:guid: train-2
I0317 12:00:49.491060 140397564315520 run_classifier.py:516] guid: train-2
INFO:tensorflow:tokens: [CLS] 偶 闻 西 地 省 m3 县 人 民 医 院 部 分 聘 用 制 员 工 因 为 所 谓 的 不 属 于 在 编 人 员 ， 不 享 有 带 薪 年 假 ， 在 如
I0317 12:00:49.491323 140397564315520 run_classifier.py:518] tokens: [CLS] 偶 闻 西 地 省 m3 县 人 民 医 院 部 分 聘 用 制 员 工 因 为 所 谓 的 不 属 于
INFO:tensorflow:input_ids: 101 981 7319 6205 1765 4689 9305 1344 782 3696 1278 7368 6956 1146 5470 4500 1169 1447 2339 1728 711 2792 6458 4638 679 2247
I0317 12:00:49.491569 140397564315520 run_classifier.py:519] input_ids: 101 981 7319 6205 1765 4689 9305 1344 782 3696 1278 7368 6956 1146 5470 4500 116
INFO:tensorflow:input_mask: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
I0317 12:00:49.491798 140397564315520 run_classifier.py:520] input_mask: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
INFO:tensorflow:segment_ids: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
I0317 12:00:49.492026 140397564315520 run_classifier.py:521] segment_ids: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
INFO:tensorflow:label: 劳动和社会保障 (id = 4)
I0317 12:00:49.492119 140397564315520 run_classifier.py:522] label: 劳动和社会保障 (id = 4)
```

图 2-6 部分训练过程

图 2-6 中 Num examples 为加载训练数据样本 6123 个，batch_size：批大小，即 1 次迭代所使用的样本量。以及对数据进行标注和向量化，并且随机 mask 后训练模型的过程。

2.4.6 模型的训练结果

$$accuracy=\frac{TP+TN}{TP+TN+FP+FN} \tag{2-1}$$

True Positive（TP）：预测为正例，实际为正例，False Positive（FP）：预测为正例，实际为负例，True Negative（TN）：预测为负例，实际为负例，False Negative（FN）：预测为负例，实际为正例。

2.5 模型评价

使用建立的留言内容分类的 BERT-wwm 模型，对测试数据的类别进行预测，得到每个测试数据属于所有类别的概率值矩阵，矩阵中概率值最大的列即为所属类别，使用 F_1-score 对分类结果进行评价。

- **查准率 precision**

$$precision=\frac{TP}{TP+FP} \tag{2-2}$$

- **查全率 recall**

$$recall=\frac{TP}{TP+FN} \tag{2-3}$$

■ **F_1-score**

通常我们使用 F1-score 对分类方法进行评价，其中 F1-score 的式子如式 2-4 所示：

$$F_1 - score = \frac{1}{n}\sum_{i=1}^{n}\frac{2P_iR_i}{P_i + R_i} \tag{2-4}$$

其中为 P_i 第 i 类的查准率，为 R_i 第 i 类的查全率。F_1-score 为 0 到 1 之间的值，越接近于 1，模型的效果越好。

True Positive（TP）：预测为正例，实际为正例，False Positive（FP）：预测为正例，实际为负例，True Negative（TN）：预测为负例，实际为负例，False Negative（FN）：预测为负例，实际为正例.

3 问题二：

3.1 问题分析

为了及时发现热点问题，让相关部门进行有针对性地处理，提升服务效率，需要做热点问题挖，根据附件 3 提供的内容，对数据进行清洗和提取特征后， 采用 **DBSCAN 聚类模型**对留言主题和留言详情的摘要进行聚类，得到多个聚类簇，通过自定义热度指数对每个聚类簇进行排名，得到前 5 个热点问题，提取时间范围和地点人群的信息，使用 **TextRank 算法**提取每一簇里面得分最高的留言主题作为该热点的问题描述，并保存为文件“热点问题表.xlsx”。最后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.xlsx”。

3.2 模型建立

3.2.1 热度评价模型

经过聚类之后，表述同一个问题的留言被聚成了一簇，形成了若干问题簇，需要判断哪些簇算得上是热点，因此我们定义一个热度指数来将其进行排名，本文考虑到的第一个指标便是反应同一个问题的留言数量，第二个便是留言的点赞数和反对数，因为点赞数和反对数同样能反应民众的心声，点赞表示对问题有着同样的看法，反对表示此问题持否定态度。因此，本文将热度指标设定为：

$$\text{热度指数}=w_1\times\text{该热点留言总数}+w_2\times\frac{\text{该热点总点赞数}-\text{该热点总反对数}}{\text{该热点留言总数}} \quad (3\text{-}1)$$

其中 w_1 和 w_2 为自定义权重，具体取值可见模型求解。

3.3 数据预处理

3.3.1 数据清洗

对留言进行去重、然后对新闻内容文本进行 jieba 分词并词性标注，过滤出名词、动词、简称等词性，分词前使用自定义的用户词词典增加分词的准确性，分词后使用停用词词典、消歧词典、保留单字词典过滤掉对话题无关并且影响聚

类准确性的词，建立每篇留言的词库。

3.3.2 特征提取

利用 TF-IDF [3]特征提取，得到每篇留言的特征向量。

TF 算法：统计一个词在一篇文档中出现的频次。基本思想是一个词在文档中出现的次数越多，则其对文档的表达能力也就越强。公式如下：

$$tf_{ij} = \frac{n_{ij}}{\sum_{k} n_{kj}} \tag{3-2}$$

IDF 算法：统计一个词在文档集的多少个文档中出现。基本的思想是一个词在越少的文档中出现，其对文档的区分能力也就越强。公式如下：

$$idf_i = \log(\frac{|D|}{1+|D_i|}) \tag{3-3}$$

TF－IDF 算法就是 TF 算法与 IDF 算法的综合，计算公式如下：

$$tf \times idf(i,j) = tf_{ij} \times idf_i = \frac{n_{ij}}{\sum_{k} n_{kj}} \times \log(\frac{|D|}{1+|D_i|}) \tag{3-4}$$

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。

3.4 算法介绍

3.4.1 DSSCAN 聚类算法

利用 TF-IDF 特征提取之后对新闻进行 DBSCAN 聚类[4]，并对每个类的大小进行排序。基于密度的 DBSCAN 算法进行新闻聚类是最为准确的做法，这种算法的特点可以发现任意形状的簇，同时可以过滤离群点，而不必把离群点分在某一个簇中，增加聚类的偏差。

DBSCAN 聚类的原理很简单：由密度可达关系导出最大密度相连的样本集合（聚类）。这样的一个集合中有一个或多个核心对象，如果只有一个核心对象，则簇中其他非核心对象都在这个核心对象的ε邻域内；如果是多个核心对象，那么任意一个核心对象的ε邻域内一定包含另一个核心对象（否则无法密度可达）。这些核心对象以及包含在它ε邻域内的所有样本构成一个类。算法流程图如下图

所示：

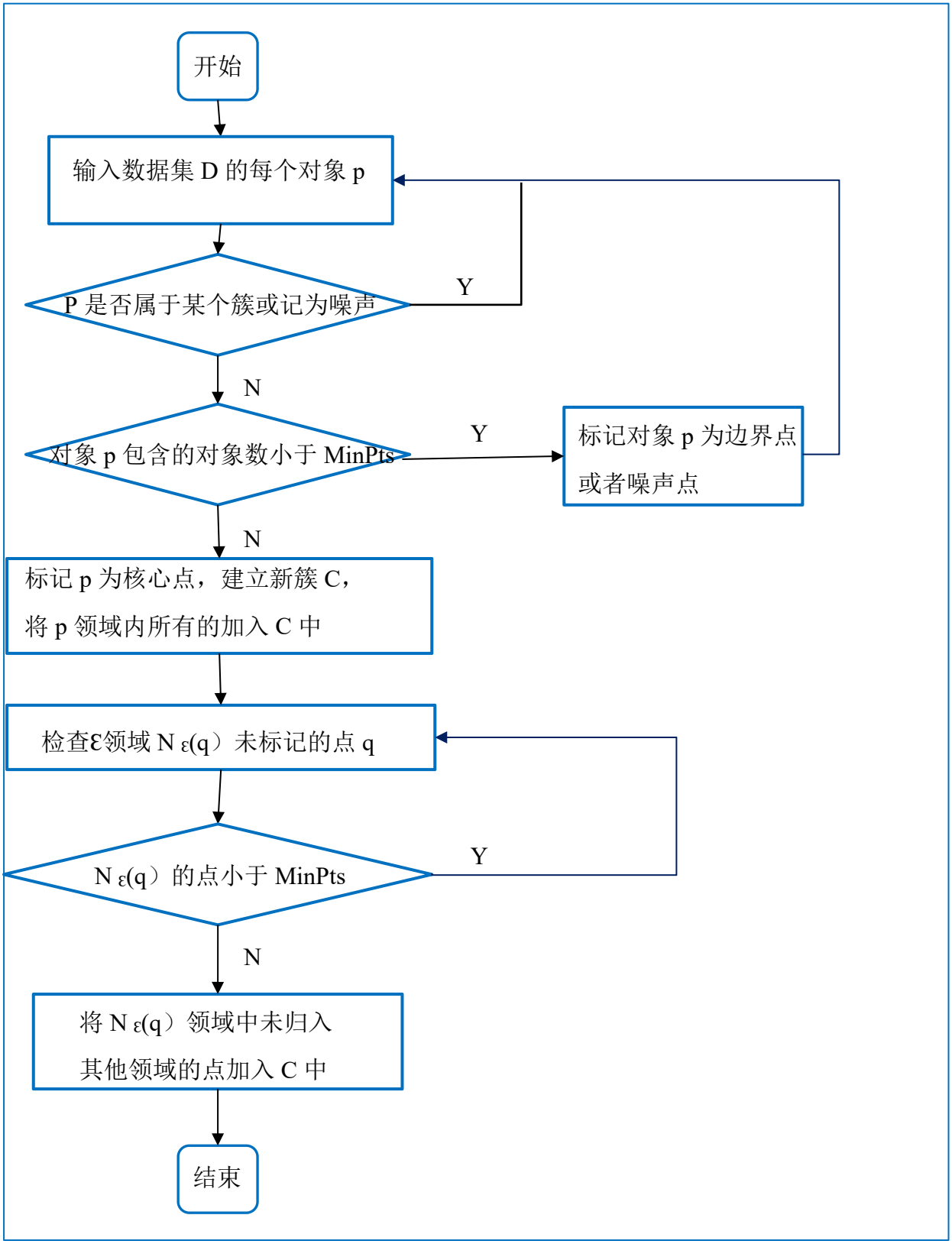

图 3-1 DBSCAN 流程图

在聚类的过程中需要输入三个参数：

（1）D：一个包含 n 个对象的数据集

（2）Ɛ：半径参数

（3）MinPts：领域密度阈值

在计算过程中，使用“余弦相似度”来计算距离，所以本文在设置ε参数时，一般都设置为 0.3-0.5 之间，因为超过 0.5 的半径值会使不属于同类新闻聚在一起，小于 0.3 则无法识别相同事件的新闻。设置 MinPts 参数时，可以人为的假定一个阈值，假设有 10 条新闻报道同一事件就认为它就是一个热点，那么可以设置 MinPts 值为 10。

3.4.2 TextRank 自动文本摘要

针对聚类后的每一类热点，为了得到该处热点的话题信息，还需要提取它们的标题，利用 TextRank 算法[5]，对标题的重要程度进行排序，用重要性最高的标题来描述该处热点的话题。

TextRank 是一种基于图的用于文本的排序算法，基本思想来自于 Google 的 PageRank 算法[3]。类似于网页的排名，对于词语可得到词语的排行，对于句子也可得到句子的排名，所以 TextRank 可以进行关键词提取，也可以进行自动文摘。其用于自动文摘时的思想是：将每个句子看成 PageRank 图中的一个节点，若两个句子之间的相似度大于设定的阈值，则认为这两个句子之间有相似联系，对应的这两个节点之间便有一条无向有权边，边的权值是相似度，接着利用 PageRank 算法即可得到句子的得分，把得分较高的句子作为文章的摘要。

TextRank 算法的主要步骤如下：

（1）预处理：分割原文本中的句子得到一个句子集合，然后对句子进行分词以及去停用词处理，筛选出候选关键词集。

（2）计算句子间的相似度：采用如下公式进行计算句子 i 和句子 j 的相似度：

$$Similarity(S_i, S_j) = \frac{n}{\log(S_i) + \log(S_j)} \tag{3-5}$$

其中，n 为两个句子都出现的词的数目，S_i 为句子 i 中的词的数目，S_j 为句子 j 中的词的数目。对于两个句子之间的相似度大于设定的阈值的两个句子节点用边连接起来，设置其边的权重为两个句子的相似度。

（3）计算句子权重：

$$WS(V_i)=(1-d)+d\times\sum_{V_j\in In(V_i)}\left(\frac{w_{ji}}{\Sigma_{V_k\in Out(V_j)}w_{jk}}\times WS(V_j)\right) \tag{3-6}$$

其中，$WS(V_i)$表示句子 i 的权重，d 为阻尼系数，w_{ji}表示句子 i 和句子 j 的相似度，$WS(V_j)$ 表示句子 j 的权重，$V_j\in In(V_i)$ 表示与句子 i 相连的句子，$\Sigma_{V_k\in Out(V_j)}w_{jk}$ 表示所有与句子 j 相连的句子的边的权重和。

（4）形成文摘：将句子按照句子得分进行倒序排序，抽取得分排序最前的几个句子作为候选文摘句，再依据字数或句子数量要求筛选出符合条件的句子组成文摘。

3.5 模型求解

3.5.1 求解流程

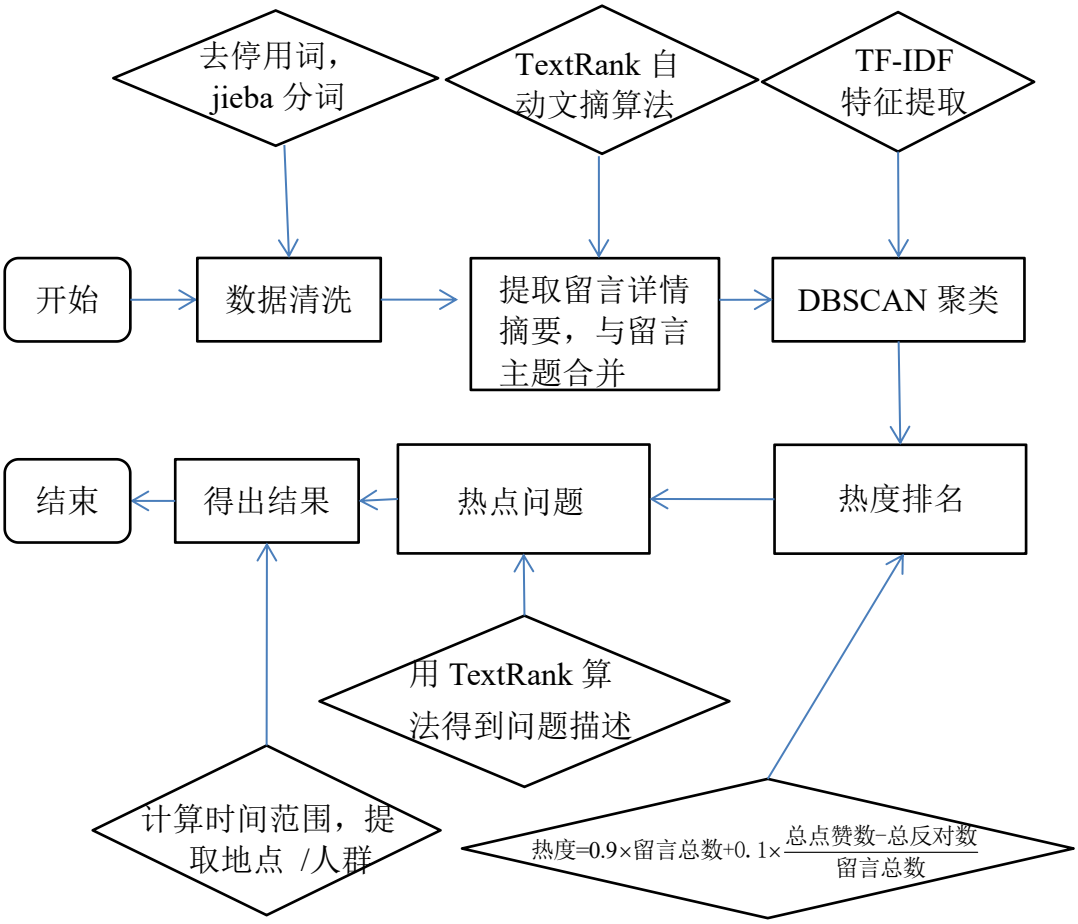

图 3-2 求解流程图

第一步：对留言内容进行清洗并进行结巴分词及词性筛选。

第二步：由于单纯使用留言主题进行聚类和单纯使用留言详情进行聚类，效果都不是很好。于是本文决定采用留言主题和留言详情相结合的方法进行聚类，使用 TextRank 自动文本摘要算法提取每条留言详情里得分最高的 3 个句子作为每条留言的摘要，并将其与留言主题合并。

这样做是因为留言主题是对留言详情的高度概括，所以要利用起来，而留言详情内容太多又会影响聚类效果，使用 TextRank 算法提取留言摘要，这样就间接地利用了留言详情，能够拥有较好的聚类效果。

第三步：使用 TF-IDF 算法提取特征，得到每篇留言的特征向量。

第四步：使用 DBSCAN 密度聚类模型根据留言主题和留言详情的摘要结合起来的文本对留言内容进行聚类，得到多个聚类簇。离群点会被筛选掉。关于 DBSCAN 密度聚类需要设置的参数：本文设置 ε 参数为 0.49，MinPts 参数为 5；一共聚成 27 个聚类簇（不包含离群点）。

第五步：通过式 3-1 所建立的热度评价模型对每个聚类簇进行排名，令其 $w_1 = 0.9$，$w_2 = 0.1$；使用 TextRank 算法提取每一簇里面得分比较高的留言主题作为该问题描述，从而得热点问题的排名。

$$\text{热度指数}=0.9\times\text{该热点留言总数}+0.1\times\frac{\text{该热点总点赞数}-\text{该热点总反对数}}{\text{该热点留言总数}} \quad (3\text{-}7)$$

4 问题三：

4.1 问题分析

针对附件 4 中的留言详情和答复意见，对答复的相关性、完整性、可解释性，及时性进行定义，建立对答复意见的质量评价模型，从而完成对答复质量的评价。

4.2 解题思路

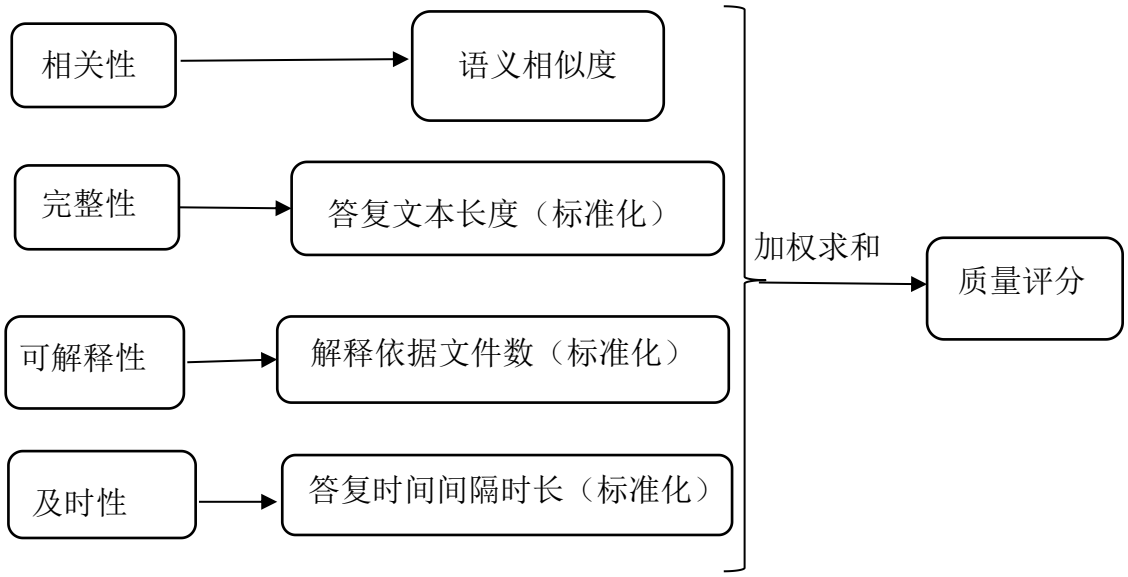

图 4-1 解题思路示意图

4.3 模型建立

4.3.1 质量评价模型

相关性：本文使用 Doc2vec 将留言详情和答复意见转化为文本向量，采用余弦距离计算文本**语义相似度**。如果它们的相似性过低，显然是答非所问或者是使用通用答复敷衍回答。故文本语义是否相关是答复意见质量评价的一个重要指标。其中，余弦距离公式：

$$\cos(\theta)=\frac{A\bullet B}{|A|\times|B|} \tag{4-1}$$

A 为留言文本向量，B 为答复意见文本向量，两个向量之间夹角的余弦值越大，语义相似度越高 。

完整性：本文采用答复意见**文本长度**作为衡量完整性的一个指标，如答复的字数太少，显然是一个应付性答案。

可解释性：答复意见文本内容里带书名号的可能是一些法律法规，政策文件等等，答复意见引用了这些文件，显得答复有理有据，可解释性较强，本文统计了答复意见文本里带书名号内容的数量即**解释依据文件数**，作为可解释性指标。

及时性：答复能够及时是问政效率的体现，老百姓渴望等到及时回复，解决急需解决的问题。及时性亦会影响答复意见的质量。本文统计了每条留言的**答复时间间隔**，作为及时性指标。公式为：

$$\text{答复时间间隔} = \text{答复时间} - \text{留言时间} \tag{4-2}$$

为了消除量纲的影响，还需对数据进行标准化，将数据转换到[0,1]之间，定义标准化公式为：

$$X_{scale} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{4-3}$$

最后**综合**以上评价指标，对他们进行加权求和，得到答复意见质量评价方案。公式为：

$$\begin{aligned} \text{质量评分} = {} & w_1 \times \text{语义相似度} + w_2 \times \text{答复文本长度(标准化)} \\ & + w_3 \times \text{解释依据文件数(标准化)} + w_4 \times \text{答复时间间隔(标准化)} \end{aligned} \tag{4-4}$$

其中 w_1、w_2、w_3、w_4 为自定义权重，具体取值可见模型求解。

4.4 数据预处理

1、将附件 4 中的留言详情和答复意见，分别提取出来，并按顺序保存为 txt 文档；

2、为避免内容中出现繁体中文，因此将繁体字转化为简体字；

3、用语料库训练词向量和测试模型之前需要对中文句子进行分词，采用 Jieba 中文分词工具对句子进行分词。

4、去停用词，避免无关符号和其他无实际意义的词对模型造成干扰，因此

需对留言详情和答复意见文本进行去停用词操作。

4.5 算法介绍

4.5.1 Doc2vec

Doc2vec[6][7]又叫 Paragraph Vector，或者叫做 paragraph2vec, sentence embeddings，是一种非监督式算法，可以获得 sentences/paragraphs/documents 的**向量表达**，是 Tomas Mikolov 在 2014 年基于 Word2vec 模型提出的，其具有一些优点，比如不用固定句子长度，接受不同长度的句子做训练样本。在一般的文本处理任务中，会将词向量和段向量相结合使用以期获得更好的效果。

Word2vec 的基本思想是：上下文相似的词，其语义也相似。根据上下文的单词来预测当前词的概率。 Word2Vec 主要有 CBOW 和 Skip-Gram 两种模型，从直观上理解，如果是用一个词语的上下文作为输入来预测这个词语本身，则是 CBOW 模型。如果是用一个词语作为输入来预测它周围的上下文，那这个模型叫做 Skip-gram 模型。

Doc2vec 增加了一个与词向量长度相等的段落向量 paragraph id，该向量具有固定长度，它不仅加入了文本语义信息，而且提取了文本的语序信息，同时具有更好的泛化能力。Doc2vec 有两种模型： Distributed Memory（DM）和 Distributed Bag Of Words(DBOW)，分别对应于 Word2vec 中的 CBOW 和 Skip-gram。DM 模型给定的上下文不仅包括上下文单词而且还包括相应的段落。DBOW 模型则在仅给定段落向量的情况下预测段落中一组随机单词的概率，与 Skip-gram 模型只给定一个词语预测目标词概率分布类似。

（1）在 DM 模型中，每一句话用唯一的向量来表示，用矩阵 D 的某一列来代表。每一个词也用唯一的向量来表示，用矩阵 W 的某一列来表示。每次从一句话中滑动采样固定长度的词，取其中一个词作预测词，其他的作输入词。输入词对应的词向量 word vector 和本句话对应的句子向量 Paragraph vector 作为输入层的输入，将本句话的向量和本次采样的词向量相加求平均或者累加构成一个新的向量 X，进而使用这个向量 X 预测此次窗口内的预测词。它的框架如图 4-2 所示：

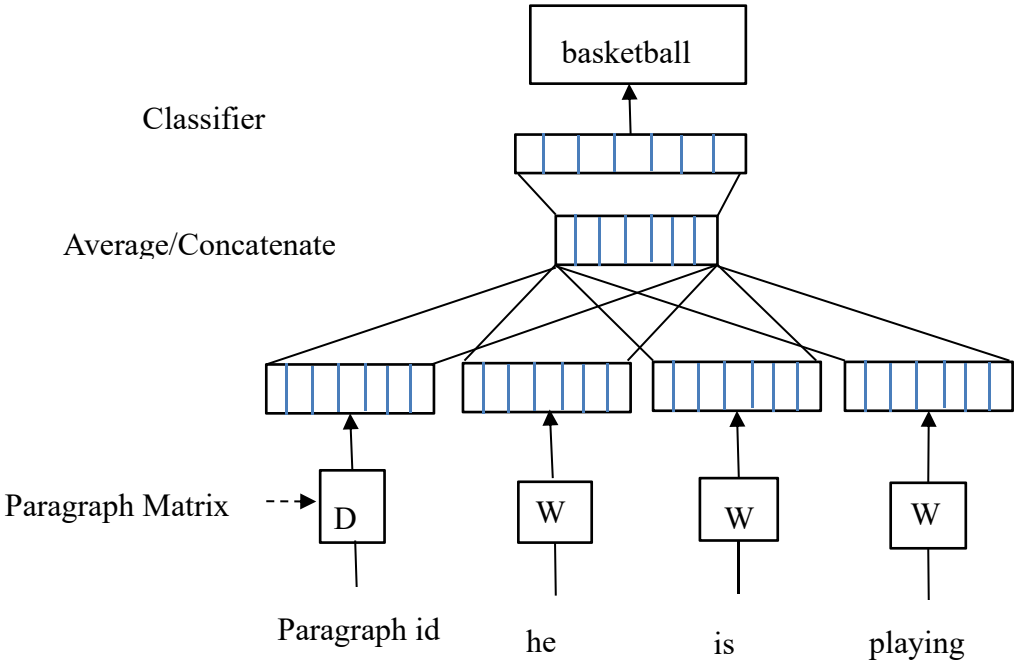

图 4-2 DM 模型示意图

（2）DBOW 模型是忽略输入的上下文，让模型去预测段落中的随机一个单词。就是在每次迭代的时候，从文本中采样得到一个窗口，再从这个窗口中随机采样一个单词作为预测任务，让模型去预测，输入就是段落向量。如图 4-3 所示：

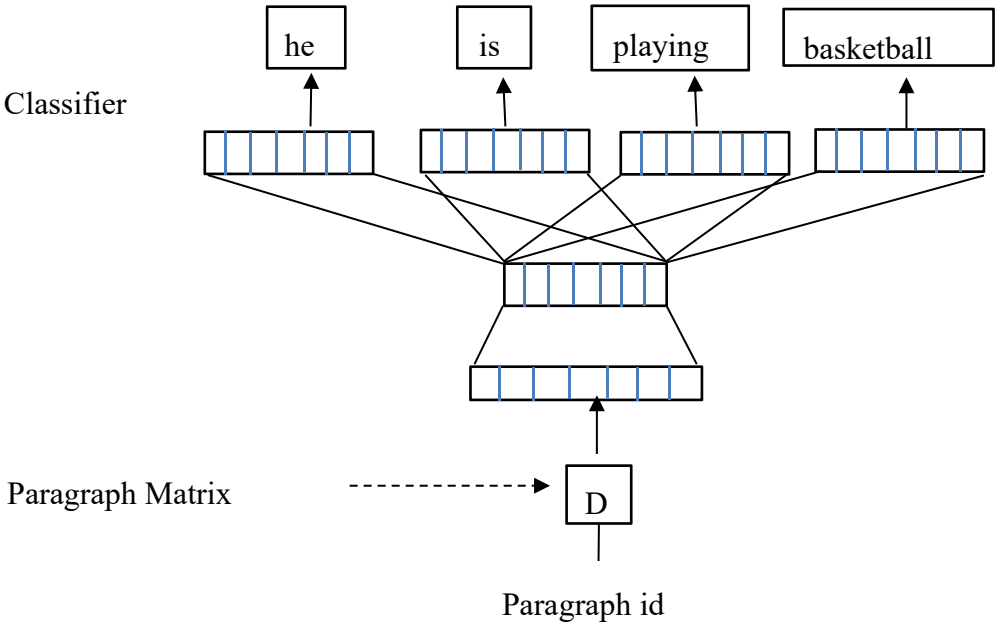

图 4-3 DBOW 模型示意图

4.6 模型求解

第一步：计算文本语义相似度。使用 Doc2vec 模型将文本进行向量化，然后计算分别每篇留言和对应答复的余弦相似度，部分结果如下：

```
第1个问答
相似度：39.22%
第2个问答
相似度：24.04%
第3个问答
相似度：37.94%
第4个问答
相似度：49.50%
第5个问答
相似度：48.97%
第6个问答
相似度：18.52%
第7个问答
相似度：44.68%
第8个问答
相似度：26.57%
第9个问答
相似度：46.96%
第10个问答
相似度：35.11%
```

图 4-4 语义相似度部分计算结果

第二步：计算文本长度，部分结果如下：

```
The file a2790.txt has about 166 words.
The file a2791.txt has about 212 words.
The file a2792.txt has about 493 words.
The file a2793.txt has about 145 words.
The file a2794.txt has about 29 words.
The file a2795.txt has about 815 words.
The file a2796.txt has about 181 words.
The file a2797.txt has about 288 words.
The file a2798.txt has about 1295 words.
The file a2799.txt has about 219 words.
The file a2800.txt has about 21 words.
The file a2801.txt has about 21 words.
The file a2802.txt has about 21 words.
The file a2803.txt has about 120 words.
The file a2804.txt has about 187 words.
The file a2805.txt has about 192 words.
The file a2806.txt has about 613 words.
The file a2807.txt has about 423 words.
The file a2808.txt has about 22 words.
The file a2809.txt has about 22 words.
The file a2810.txt has about 231 words.
The file a2811.txt has about 33 words.
The file a2812.txt has about 39 words.
The file a2813.txt has about 637 words.
The file a2814.txt has about 498 words.
The file a2815.txt has about 244 words.
```

图 4-5 文本长度部分计算结果

第三步：计算解释依据文件数。使用 python 中的 re 模块，运用正则表达式将每条答复意见里带书名号的内容提取出来后统计数量，部分结果如下：

```
The file a2795.txt 引用了:['《问政西地省》', '《燃烧器适配性目录》']数量:2
The file a2796.txt 引用了:['《H3县许家坊乡政府不顾反对\n强行在居民楼前修厕所》']数量:1
The file a2797.txt 引用了:['《问政西地省》', '《住房公积金管理条例》']数量:2
The file a2798.txt 引用了:['《问政西地省》', '《(H3县出租汽车行业创“三优”竞赛活动实施方案》', '《出租汽车客运市场经营秩序专项整治行动实施方案》', '《网络预约出租汽车经营服务管理暂行办法》', '《网络预约出租汽车经营许可证》', '《网络预约出租汽车运输证》', '《网络预约出租汽车驾驶员证》', '《网络预约出租汽车经营服务管理实施细则》', '《网络预约出租汽车经营服务管理实施细则》']数量:9
The file a2799.txt 引用了:[]数量:0
The file a2800.txt 引用了:[]数量:0
The file a2801.txt 引用了:[]数量:0
The file a2802.txt 引用了:[]数量:0
The file a2803.txt 引用了:[]数量:0
The file a2804.txt 引用了:[]数量:0
The file a2805.txt 引用了:[]数量:0
The file a2806.txt 引用了:[]数量:0
The file a2807.txt 引用了:['《问政西地省》', '《GB-5768-2009》']数量:2
The file a2808.txt 引用了:[]数量:0
The file a2809.txt 引用了:[]数量:0
The file a2810.txt 引用了:['《问政西地省-刘安革专栏》']数量:1
The file a2811.txt 引用了:[]数量:0
The file a2812.txt 引用了:[]数量:0
The file a2813.txt 引用了:['《致家长一封信》', '《致家长一封信》']数量:2
The file a2814.txt 引用了:['《问政西地省》']数量:1
The file a2815.txt 引用了:[]数量:0
```

图 4-6 解释依据文件数部分计算结果

第四步：计算答复时间间隔。使用 python 中 pandas 模块的 to_datetime()函数将答复时间和留言时间换成标准时间格式然后进行相减，部分结果如表 4-1：

表 4-1 答复时间间隔部分计算结果

Index	时间间隔
0	15 days 05:24:44
1	14 days 17:45:30
2	14 days 18:09:10
3	14 days 18:42:12
4	15 days 16:48:11
5	31 days 01:24:48
6	40 days 22:25:35
7	28 days 12:31:01
8	16 days 05:34:43
9	16 days 05:45:06
10	70 days 17:36:03
11	30 days 11:24:10

第五步：为消除量纲的影响，对文本长度，解释依据文件数，答复时间间隔（转化为秒）这些指标，根据式 4-3 进行标准化，将结果转化到[0,1]之间。

第六步：将语义相似度，文本长度（标准化），解释依据文件数（标准化），答复时间间隔时长（标准化）按照式 4-4 进行加权求和，令其 $w_1=0.4$，$w_2=0.3$，$w_3=0.1$，$w_4=-0.2$ 得到综合评价指标——**质量评分**

$$\begin{aligned}\text{质量评分}=&0.4\times\text{语义相似度}+0.3\times\text{答复文本长度(标准化)}\\&+0.1\times\text{解释依据文件数(标准化)}-0.2\times\text{答复时间间隔(标准化)}\end{aligned}\tag{4-5}$$

其频数分布直方图如图 4-7 所示：

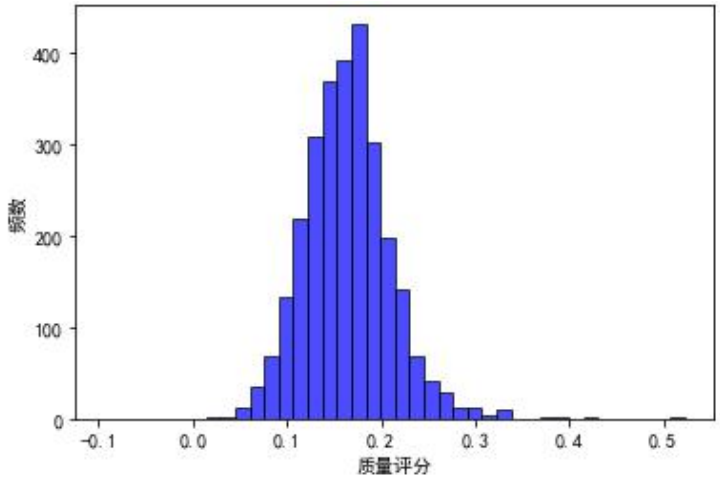

图 4-7 质量评分的频数分布直方图

最后设定阈值，将质量评分大于 0.2 打上“高质量”的标签，质量评分小于 0.1 的打上“低质量”的标签。质量评分在 0.1~0.2 之间的打上“中等质量”的标签部分结果如下：

参考文献

[1] DEVLIN J,CHANG M W,LEE K,et al. Bert:Pre-training of deep bidirectional transformers for languag e understanding[J/OL]. 2018,arXiv:1810. 04805,(2018-10-11)[2019-06-01]. https:/arxiv.org/abs/1810.04805
[2] VASWANI A,SHAZEER N,PARMAR N,et al. Attention is all you need[C]//Advances in Neural Information Processing Systems. 2017:5998-6008.
[3] 涂铭、刘祥、刘树春.Python 自然语言处理实战核心技术与算法[M].机械工业出版社，86-91
[4] Clustering 聚类 密度聚类——DBSCAN[z].https://www.cnblogs.com/PJQOOO/p/11838288.html,2020.4.4
[5] 叶建成.利用文本挖掘技术进行新闻热点关注问题分析[D].广州：广州大学，2018：14-20
[6] Benabderrahmane S，Mellouli N，Lamolle M.On the predictive analysis of behavioral massive job data using embedded clustering and deep recurrent neural networks[J].Knowledge-Based Systems，2018，151：95-113.
[7] Johnson0722.doc2vec 原理及实践[z].https://blog.csdn.net/John_xyz/article/details/79208564,2020.4.23