自然语言处理大作业

题　　目：　基于 BERT 的留言文本分类

学　　院：　计算机科学与工程学院

年　　级：　2022 级

姓　　名：　汪柏乐

学　　号：　2201867

2023 年 01 月 11 日

摘要

近年来，随着 AI 技术的发展，互联网在日常生活中处于越来越重要的地位，互联网上公众舆论的发生率也持续呈上升趋势。对网络舆论进行监督与分类已日益重要，但是手动处理方法存在诸如工作量大，低效率和高错误率之类的问题。

因此，本文采用了一种基于 BERT 的公众舆论分类框架。这是一种利用日益完善的人工智能技术来实现公众舆论分类的一种新尝试。

关键词

数据挖掘；文本分类；BERT；FARM；

目录

第一章 绪论

1.1 研究背景及意义

1762 年的时候，一位伟人卢梭就曾这样概括过舆情，所谓舆情，就是“公众”和“意见”的结合体，舆情无非是人民群众对公众事物的意见或者看法。舆情可以小到是人民群众生活中对点滴琐碎的诉说，也可以大到是对国家的发展以及社会经济进步的讨论。舆情可以说自古以来就发挥着巨大的作用，特别是在 2022 年的今天，几乎全国人民都可以在网上发出自己的声音，这也是本文要进行舆情研究分类的原因。

随着互联网在全球范围内的飞速发展，根据中国互联网络信息中心(CNNIC)发布的第 49 次《中国互联网络发展状况统计报告》显示，截至 2021 年 12 月，我国网民规模为 10.32 亿人，较 2020 年 12 月增长 4296 万，互联网普及率达 73.0%[1]。具体增长比率如下图 1-1 所示：

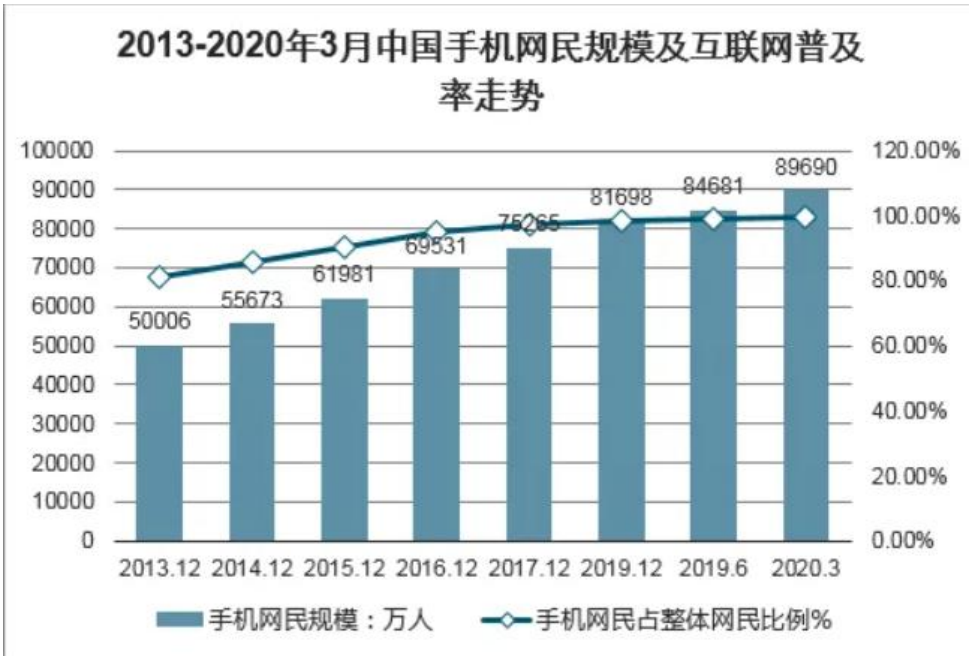

图 1-1 网民规模及互联网普及率[9]

目前微博、短视频领域、知乎以及博客涌现出了大量信息。从 2019 年疫情爆发以来，越来越多人被迫居家隔离，在家里上网课或者办公，而这也反哺了网络世界的发展。各类社交媒体以及网络平台日用户量发生了激增，各省官员逐渐开始在网上与人民群众沟通，中国官员利用互联网收集有关政府的建议，人民群众也能通过问政平台监督、投诉违规违法行为，进一步促进政府信息的透明畅通。。

时代随着历史的长河在不断发展，往日低效、落后的人工筛选评论的做法注定将被遗弃。随着智慧城市，智慧交通等一系列高科技方案被提出，智慧舆情分析分类的出现也变得迫在眉睫。让科技融于生活，让科技改善民生，让蓬勃发展的 AI 技术落至实地，解放低效生产力是新时代政府的一个重要子课题，利用 AI 技术高效、准确地处理海量文本，有助于提升行政效能，提高政府的服务质量，增强人民群众的幸福感和满意度，维护社会的稳定，这也是本文的研究方向。

1.2 国内外研究现状

与以往舆论分析的情感识别不同，由于本次研究是针对在线问政平台的留言，所以此次研究重点是文本分类。

借鉴于视觉的预训练模式，自然语言处理于 03 年提出了早期的 Word Embedding 预训练技术[2]。而预训练技术与语言模型是密不可分的，一般来说，都是在语言模型上进行大量数据的预训练，然后再在下游任务进行相应的调整。

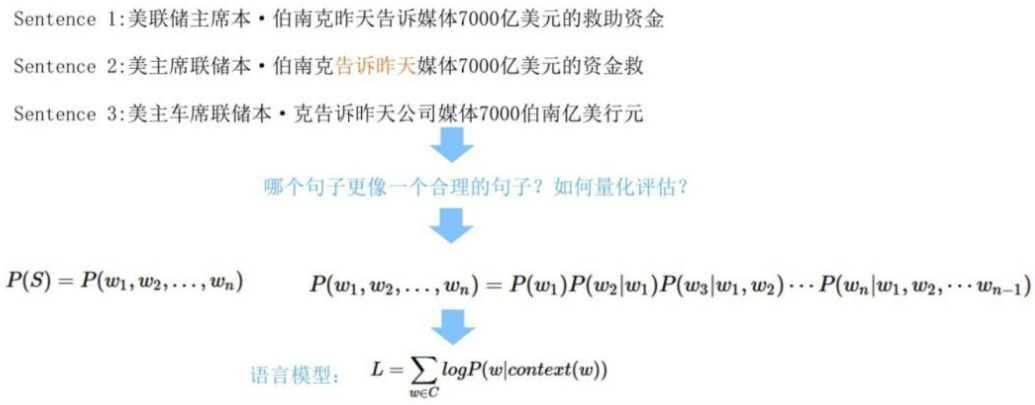

图 1-2 语言模型简要介绍

如上图 1-2 所示，语言模型其实是为了更好地判断哪个句子是一个完整的符合人类语言习惯的句子。函数 P 代表通过前面的词预测后面词的概率大小(也可以是根据上下文预测当前词的概率大小)，句中每个单词都会进行判断，所有单词概率乘起来就是这个句子的概率，句子概率越大越符合人类语言习惯。

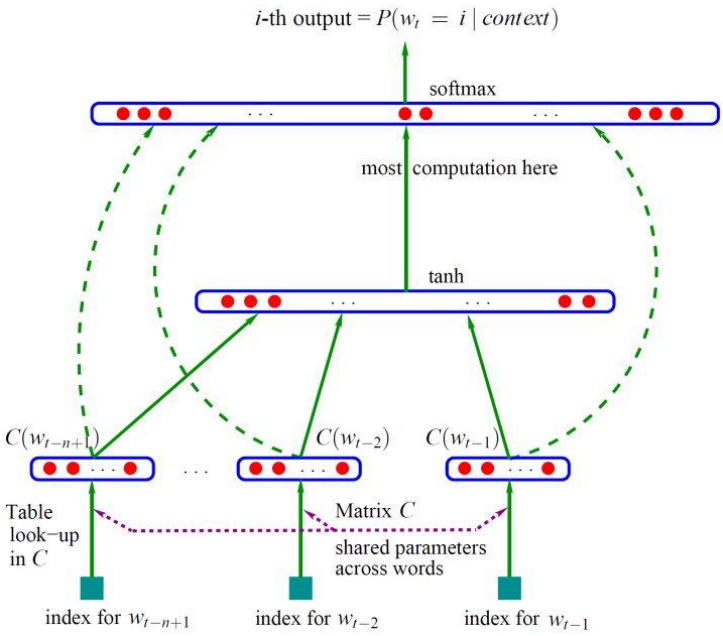

图 1-3 NNLM 模型图

如上图 1-3 所示，这是 Bengio 在 2003 年在 JMLR 上发表的 NNLM 网络结构[2]。它虽然在 2003 就被提出了，但是真正流行起来是在 2013 年，13 年其被考古选入 NLP，开创了深度学习与 NLP 结合的光辉时代，打响了用深度学习进行自然语言处理的第一枪。

在现在看来，NNLM 这个网络结构可能会略显简陋。学习任务是输入某个句中单词W_t ="dog"前面句子的 t-1 个单词，要求网络正确预测单词 dog，即最大化公式(1-1)：

$$\mathrm{P}(W_t = "dog" | W_1, W_2, ...W_{(t-1)}; \vartheta) \tag{1-1}$$

此模型的原始单词输入就是前面任意单词 W_t 通过 One-Hot 编码转换成的向量，之后乘以矩阵 Q 后获得向量 $C(W_i)$ 每个单词的 $C(W_i)$ 拼接，上接隐层，然后接 softmax 去预测后面应该后续接哪个单词。这个 $C(W_i)$ 就是单词对应的 Word Embedding 值，那个矩阵 Q 包含 V 行，V 代表词典大小，每一行内容代表对应单词的 Word Embedding 值，只不过 Q 的内容也是网络参数，需要学习获得，训练刚开始用随机值初始化矩阵 Q，当这个网络训练好之后，矩阵 Q 的内容被正确赋值，每一行代表一个单词对应的 Word Embedding 值[2]。

2013 年自 Word2Vec 被提出后，逐渐成为最热门的单词映射工具。其模型图如图 1-4 所示：

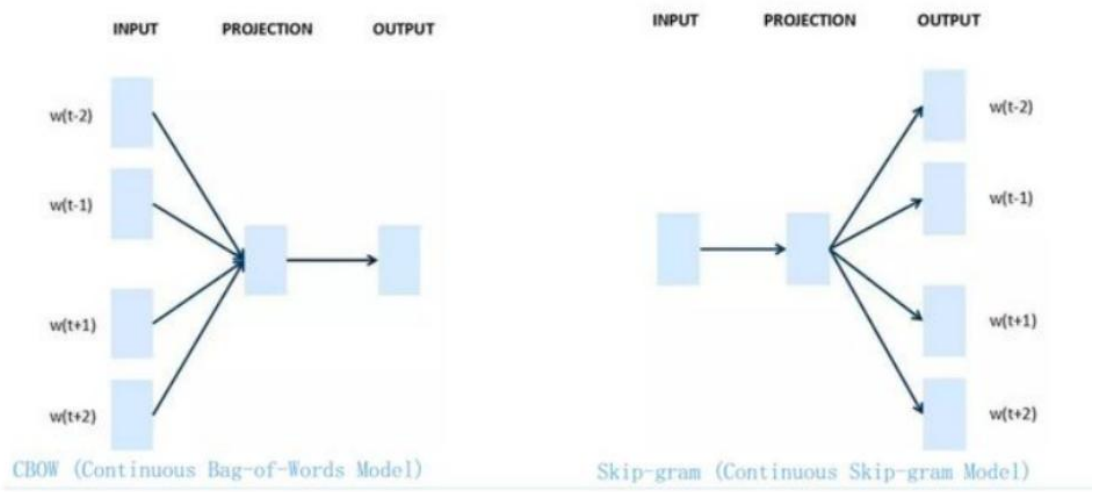

图 1-4 Word2Vec 模型图

Word2Vec 除了训练方法与 NNLM 不一样外，结构基本是差不多的。Word2Vec 有两种训练方法，一种叫 CBOW，核心思想是从一个句子里面把一个词抠掉，用这个词的上文和下文去预测被抠掉的这个词，第二种叫做 Skip-gram，和 CBOW 正好反过来，输入某个单词，要求网络预测它的上下文单词[3]。

但单词嵌入有个致命的缺陷就是一词多意的问题，由于语言的高灵活性，往往一个单词或者一个字在不同的语义中是有不同意义的，而单词嵌入(Word Embedding)每个单词对应的是一行固定的数字，是矩阵中固定的那一行，这就造成了很多情况下模型对单词的误解问题。

为了解决这个问题，ELMo[4]于 18 年横空出世，ELMO 的本质思想是：创建出一个能动态调整的 Embedding。首先得出单词的 Word Embedding，然后在下游任务中把单词的 Word Embedding 以及包含句法和语义的 Embedding 当成一个整体作为输入，这样一来，上下文语义不同，单词综合的 Embedding 也不同，就能很好地解决单词多种语义的问题。

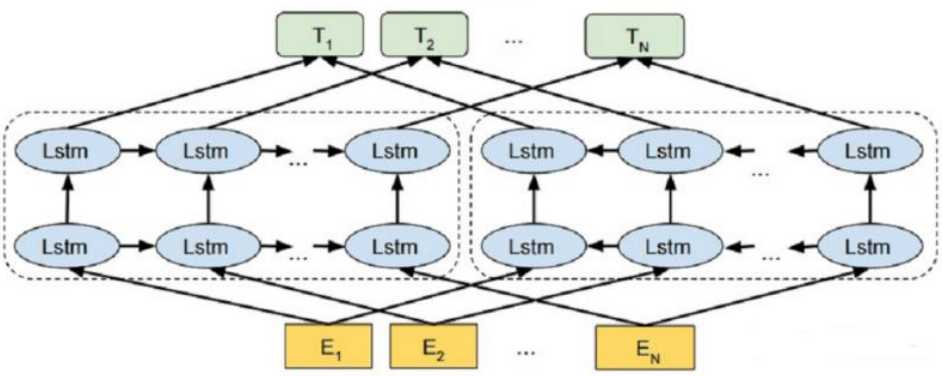

图 1-5 ELMo 模型图

ELMO 是一个经典的两步走策略，首先利用 LM(Language Model)进行预训练；接着从预训练网络中提取单词各层综合的 Embedding 成为一个新的参数补充到下游任务中。图 1-5 展示的是其预训练过程，如图所示，ELMo 采用了一个双向双层 LSTM 来构成网络结构，左边的是从左至右的正向编码器，输入的是从左到右顺序的除了预测单词外的上文，右端的逆向双层 LSTM 代表反方向编码器，输入的是从右到左的逆序的句子下文，每个编码器的深度都是两层 LSTM 叠加[4]。当我们输入一个新的句子的时候，每个单词除了最底层的单词嵌入(Word Embedding)外，还能另外通过双层 LSTM 获得两个 Embedding，一个专门针对句法信息，一个专门针对语义信息，此三者进行加权累加最后的结果才是下游任务中的输入。

历史的车轮总是不断前行的，解决完多义词的问题后，我们又将投入到如何提升网络模型效率的事情上去。ELMo 虽说很好地解决多义词的问题，但其特征抽取器的能力有很大不足。17 年 NLP 领域的大杀器 Transformer 被提出，这个自然语言处理领域的新贵一经提出，就得到了一致好评，并以其为基底衍生出了一系列新的网络模型，GPT 就是其一，其结构图如图 1-6 所示：

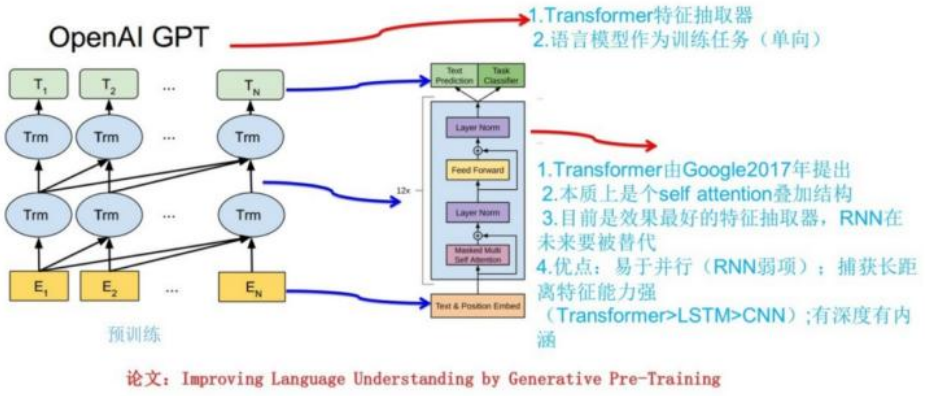

图 1-6 GPT 模型结构图

如果说 ELMo 是 feature-based 预训练方法的代表，那么 GPT[5]就是基于微调模式的典型开拓者。同为预训练网络，GPT 选择的模型架构却有很大的不同，其特征抽取器摒弃了以往的 CNN 以及 RNN，而采取了 Transformer 模型。Transformer 在长距离特征捕获方面与 CNN、RNN 相比有着天然的优势，这也促使 Transformer 成为了序列关系

捕获的主流 NLP 工具。与 BERT 同属微调的预训练模型，同时又同样采用了 Transformer，那么二者为何差距这么大呢。究其原因，还是 GPT 模型结构的原因，如图 1-6 所示，GPT 是一条单向的语言模型，单向的语言模型只能兼顾单词的上文，而无法利用单词下文的数据信息，这无疑是一种极大的浪费，倘若 GPT 也同样采用了双向的语言模型，可能其与 BERT 相争的结果就未可知了。

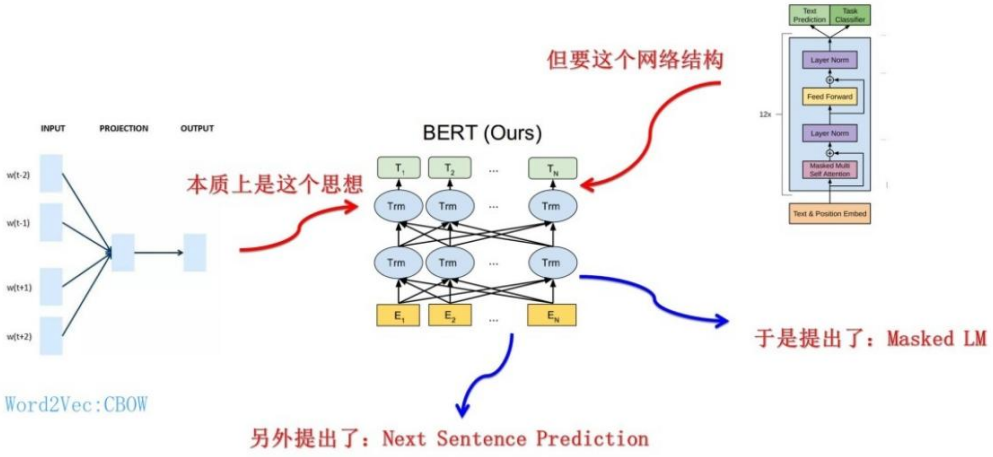

图 1-7 BERT 的模型图

如图 1-7 所示，与其说 BERT 模型是一个横空出世的全新类模型，不如说他是一个 NLP 领域模型的集大成者，融合了前人众多模型的优势。根据图 1-7 可以清晰地看到，BERT 模型主要是提出了 Masked language model 以及 Next Sentence Prediction。按照前文 CBOW 方法，BERT 的 Masked 模型做出了相应的改进。不再是将要预测的单词抹掉，通过上下文预测这个单词，而是随机抹掉语料中 15%的单词，被抹掉的单词中 80%被替换成[mask]标记，10%被随机替换成另外一个单词，10%保持不变[6]。而 Next Sentence Prediction 指的是判断第二个句子是不是接在第一个句子后的，个人感觉这个功能没有太大的影响，只有某些特殊的场景任务能用上，但这并不影响 BERT 模型在 NLP 领域各类任务上大杀四方的显著效果以及极强的普适性。

由于 BERT 其优秀的长距离特征抽取能力，以及融合一体式的双向结构，可以预见的是，未来 BERT 将在 NLP 领域势不可挡地撑起大旗。这也是本文选择用 BERT 进行相关研究的原因所在。

第二章 基于 BERT 的文本分类

2.1 数据预处理

2.1.1 数据描述

本文数据集采用的是某数据挖掘比赛政务留言的数据集，观察所给数据，发现共有 9210 个样例，每个样例由 5 个属性描述(留言编号，留言用户，留言主题，留言时间，留言详情)和 1 个标签(一级标签)组成，详情可见图 2-1。而属性(留言详情)所对应的属性值中有大量的换行符和制表符以及没有意义的语句，如果不做处理会对后续分析造成影响。于是首先要对数据进行预处理。

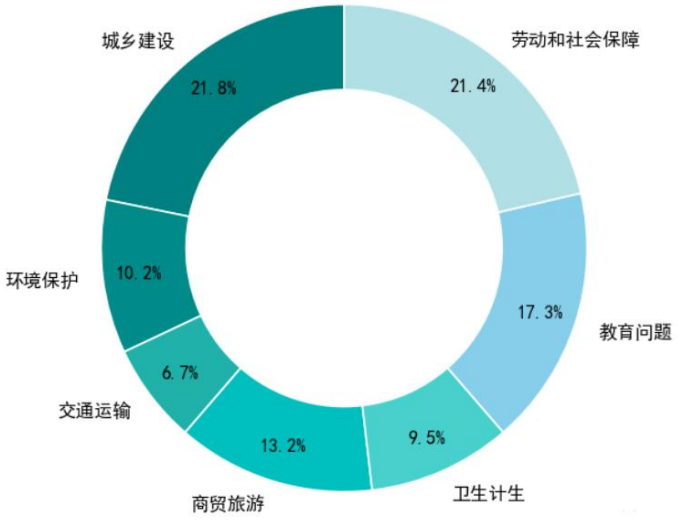

图 2-1 数据集可视化

2.1.2 文本预处理

(一)构建新的属性

由于留言主题的概括性强但留言详情又包含很多细节，于是将留言主题与留言详情拼合成新的属性(留言)。

(二)文本删减与去空

对于留言所对应的属性值，例如： '\n\t\t\t\t\n\t\t\t\tA3 区大道西行便道，未管所路口至加油 站路段，人行道包括路灯杆，被圈西湖建筑集团燕子山安置房项目施 工围墙内。每天尤其上下班期间这条路上人流车流极多，安全隐患非常大。强烈请求文明城

市 A 市，尽快整改这个极不文明的路段。 \n\t\t\t\t\t\n\t\t\t\t\t\n\t\t\t\t\t' 经过测试得出数字和英文字母对分类结果并没有积极意义，于是将数字和字母 与'\t'、"\n'、'\u3000'以及空格、标点符号去除。

2.2 BERT 的文本向量化介绍

由于计算机并不能很好地识别人类所用的文字，因此我们需要将文字转换为向量模式。常用的向量转化模式有 one-hot 向量，但此向量往往容易忽略单词的位置信息，且容易造成维度灾难。后来陆续涌现除了一大堆基于深度表示的模型，例如 Word2Vec，GloVe 模型等。虽然以上模型可以利用上下文信息预测词向量使得生成的词向量包含了寓意信息，但以往的模型有着自己的致命弊端。有的只能构建单向语言模型，有的无法解决一词多义的问题，有的在其特征抽取能力上有着天然的缺陷。针对以上问题，本文引入动态词向量 BERT 模型，BERT 模型利用其独特的 Transformer 结构对文本进行双向学习和处理，利用 self-attention 学习词间关系，使得词向量的表示能够融入句子级的语义信息，从而解决词向量无法表示一词多义的情况。

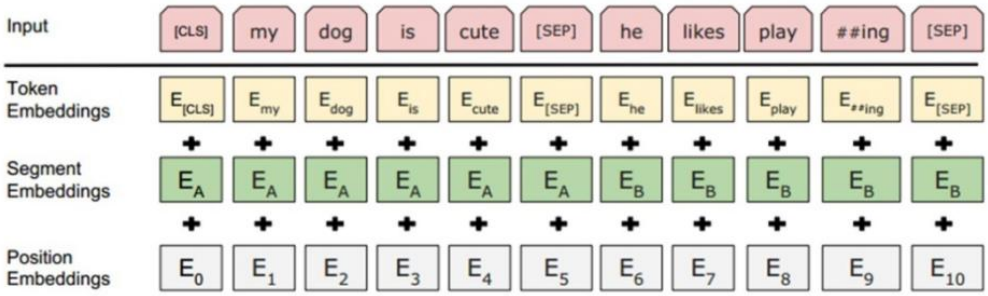

图 2-2 BERT 模型的输入部分示例

BERT 的输入是个典型的线性序列，由图 2-2 可以看出，BERT 输入的每个句子都会在句首和句尾增加两个标记符，这些标记具有特殊的意义。BERT 由 12 个 Transformer 层组成，每个 Transformer 都会接收一个标记嵌入的列表，并在输出中产生相同数量的嵌入[6]。在最后一个(第 12 个) Transformer 的输出端，分类器只使用第一个嵌入(对应[CLS]标记)。由于模型 BERT 已经被激励将分类步骤所需的一切编码到那个单一的 768 值嵌入向量中，我们将只使用这个[CLS]标记进行分类。假设输入的文本是“我是一个大学生”，BERT 将会给对句子进行相应的符号添加，变成[CLS]我是一个大学生[SEP]，进过相应的处理后每个单词会形成三个 Embedding，第一个是位置

Embedding，代表单词在句子中的位置，词语的位置顺序在 NLP 领域有很重要的意义；第二个是单词本身的 Embedding，也就是通常提到的 Word Embedding；最后一个是句子 Embedding，由于 BERT 的 NSP 功能，BERT 通常不只是单个句子进行训练，因此有个专属的句子 Embedding。此三类叠加处理后就融合成了 BERT 的输入部分。

我们数据集中的句子显然有不同的长度，面对这种情况，BERT 有两个约束条件：1. 所有的句子必须被填充或截断成一个固定的长度。 2. 最大的句子长度是 512 个 tokens。本文此处采用 512 个字符为句子截断长度，以求最好地实现 BERT 的性能。对于长度不够的句子，将会自动填充至最大长度。填充是通过一个特殊的"[PAD]"令牌来完成的，它在 BERT 词汇表中的索引 0。

2.3 通过 FARM 框架调用 BERT 模型进行分类

2.3.1 FARM 简介

FARM 框架全称是 Framework for Adapting Representation Models，它可以调用 BERT 使得迁移学习更加简单、快速，并且更加支持企业开发，它是一种基于 transformer 的框架，快速进行文本分类，NER，智能问答等任务，并轻松将这些功能应用到开发中[7]。

2.3.2 实验过程

```
label_list =["城乡建设","卫生计生","商贸旅游","劳动和社会保障","教育文体","交通运输","环境保护"]
metric = "acc"

processor = TextClassificationProcessor(tokenizer=tokenizer,
                                        max_seq_len=512, # BERT can only handle sequence lengths of up to 512
                                        data_dir='BERT留言分类数据集',
                                        label_list=label_list,
                                        label_column_name="label",
                                        metric=metric,
                                        quote_char='"',
                                        multilabel=True,
                                        train_filename="train.tsv",
                                        dev_filename=None,
                                        test_filename="test.tsv",
                                        dev_split=0.1 # this will extract 10% of the train set to create a dev set
                                        )

# 3. Create a DataSilo that loads several datasets (train/dev/test), provides DataLoaders for them and calculates a few descriptive statistics of our datasets
data_silo = DataSilo(
        processor=processor,
        batch_size=batch_size)

# 4. Create an AdaptiveModel
# a) which consists of a pretrained language model as a basis
language_model = LanguageModel.load(lang_model)

# language_model = Roberta.load(lang_model)
# b) and a prediction head on top that is suited for our task => Text classification
prediction_head = MultiLabelTextClassificationHead(num_labels=len(label_list))

model = AdaptiveModel(
```

图 2-3 通过 FARM 调用 BERT 进行分类

通过 FARM 框架调用 BERT 模型进行文本分类，数据集按照 8:1:1 的比例划分为训练集，测试集以及验证集。

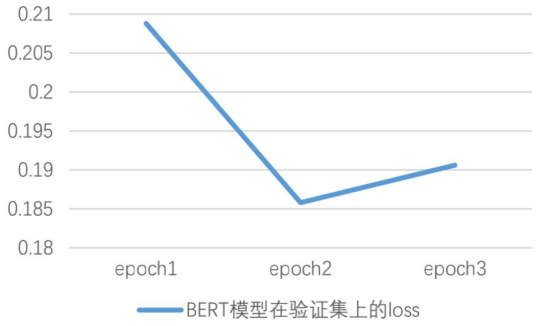

图 2-5(a) BERT 训练效果图 1

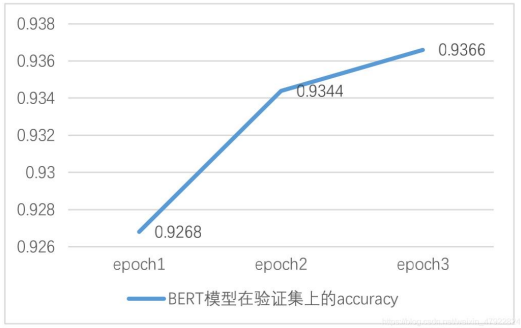

图 2-5(b) BERT 训练效果图 2

如图 2-5(a)以及图 2-5(b)所示，在第二轮 epoch 损失值达到了最低，同时在下一轮 epoch 中 accuracy 的提升幅度并不大，综合考虑，本文模型决定采用第二轮结束后的模型。

2.3.3 结果展示

最终模型在测试集上的效果如表 2-1 所示：

表 2-1 结果展示表

留言类别	查准率 P	查全率 R	F1 值
城乡建设	95.68%	88.50%	91.25%
卫生计生	89.77%	96.34%	92.94%
商贸旅游	88.19%	91.80%	89.96%
劳动和社会保障	96.89%	94.92%	95.90%
教育文体	95.63%	96.84%	96.23%
交通运输	82.61%	91.94%	87.02%
环境保护	96.77%	95.74%	96.26%

结论

自 03 年起，深度学习将近二十年如火如荼的发展引起了社会的巨大变革，一系列人工智能技术落至实地，与各行各业发展结合起来，促进了社会各行各业技术的又一次井喷式爆发。人脸识别，无人驾驶等先进技术无一不令人惊叹；语音识别，医疗机器人等便民科技也随处可见。值此时代，文本分类等传统技术也应该接收新一轮的血液注入，更好地为民服务。

通过对深度学习神经网络结构的学习，本文完成了本次对舆论留言的分类。为了解决低效、繁琐的文本分类以及人工筛选耗时耗力的问题，本文将 BERT 模型带入了留言分类，并且通过 FARM 框架调用 BERT 完成了文本的分类。

当然本文还有许多需要改进的地方，例如分类时，由于时间原因，没有对 BERT 与其余的大量分类模型进行一个对比试验等。如果未来还有时间的话，我将继续对 BERT 模型的后续优化以及应用进行一个跟进与探索，另外可以研究下最近很火的 chatGPT 的相关应用。

参考文献

[1]http://nic.upc.edu.cn/2022/0308/c7404a363798/page.htm 第 49 次《中国互联网络发展状况统计报告》

[2]Bengio Y, Ducharme R, Vincent P. A neural probabilistic language model[J]. Advances in Neural Information Processing Systems, 2003, 13.

[3]Church K W. Word2Vec[J]. Natural Language Engineering, 2017, 23(1): 155-162.

[4]Ilić S, Marrese-Taylor E, Balazs J A, et al. Deep contextualized word representations for detecting sarcasm and irony[J]. arXiv preprint arXiv:1809.09795, 2018.

[5]Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.

[6]Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[7]https://github.com/deepset-ai/FARM

附录一

```
label_list =["城乡建设","卫生计生","商贸旅游","劳动和社会保障","教育文体","交通运输","环境保护"]
metric = "acc"

processor = TextClassificationProcessor(tokenizer=tokenizer,
                                        max_seq_len=512, # BERT can only handle sequence lengths of up to 512
                                        data_dir='BERT留言分类数据集',
                                        label_list=label_list,
                                        label_column_name="label",
                                        metric=metric,
                                        quote_char='"',
                                        multilabel=True,
                                        train_filename="train.tsv",
                                        dev_filename=None,
                                        test_filename="test.tsv",
                                        dev_split=0.1 # this will extract 10% of the train set to create a dev set
                                        )

# 3. Create a DataSilo that loads several datasets (train/dev/test), provides DataLoaders for them and calculates a few descriptive statistics of our datasets
data_silo = DataSilo(
        processor=processor,
        batch_size=batch_size)

# 4. Create an AdaptiveModel
# a) which consists of a pretrained language model as a basis
language_model = LanguageModel.load(lang_model)

# language_model = Roberta.load(lang_model)
# b) and a prediction head on top that is suited for our task => Text classification
prediction_head = MultiLabelTextClassificationHead(num_labels=len(label_list))

model = AdaptiveModel(
```