上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

BACHELOR'S THESIS

论文题目：舆情文本多分类与敏感性检测技术研究与实现

学生姓名：周笑

学生学号：516030910370

专　　业：软件工程

指导教师：唐新怀

学院(系)：电子信息与电气工程学院

舆情文本多分类与敏感性检测技术研究与实现

摘要

网络环境已经成为了舆情主要来源。近些年网络舆情事件层出不穷。民众对社会事件的发生、发展和变化持有不断变化的态度。为适应新形势下的网络安全工作需要，舆情监控在实际应用中正在起到越来越重要的作用。舆情系统是能够自动采集并分析网络舆情数据的工具。系统使用大规模分布式爬虫进行全网多源舆情数据的自动化采集，然后多样化的处理模块对原始文本数据进行处理分析，为用户提供各种报表、视图与总结，帮助用户全方位的掌控舆情事件的进展。分析事件情感导向是舆情系统中极为关键的一环，情感分析的粒度与准确度直接关系着用户对事件的理解。舆情监控平台所依靠的主要技术之一就是文本分类技术。

本文基于舆情系统，利用 BERT（Bidirectional Encoder Representation from Transformers）模型，对舆情平台所需要的，针对短文本和长文本的文本多分类任务进行探究。实验采用新闻类文本作为数据集，涵盖 14 个长文本类别和 15 个短文本类别。为了考虑过长的样例文本超出了 BERT 模型能够处理的最大文本长度，本文设计了前截断，后截断，前后截断，文本摘要和拆分五种方法，探究最适合舆情文本的过长文本处理方法。实验结果表明，前后截断是更适合本实验所采用的数据集的处理方法，取得了超越当前主流模型的表现。

本文还在文本多分类的基础上，研究了敏感词检测功能在舆情平台中的实际应用。本文采用 DFA（Deterministic Finite Automaton）算法，对四种不同类别的敏感词进行快速准确的检测。最后，本文结合上述的文本多分类模型，实现了舆情监控平台的部分功能模块，并结合舆情平台将这些功能进行展示与模拟。

关键词：舆情系统，文本多分类，BERT 模型，敏感词检测，DFA 算法

THE RESEARCH AND REALIZATION OF MULTI-LABEL CLASSIFICATION AND SENSITIVITY-DETECTION TECHNOLOGY OF PUBLIC VOICE TEXTS

ABSTRACT

The network environment has become the main source of public voicie. In recent years, public voicing events on the network are emerging and people have always been changing attitudes towards the generation and developments of society. In order to meet the needs of network security under the new condition, public voice monitoring is playing an increasingly important role in practical application. Public voicing system is an instrument that can automatically collect and analyze public voice data on the network. The system utilizes large-scale distributed crawlers to automatically collect multi-source public voice data. Then the diversified processing modules commence to process and analyze the original text data, providing users with a variety of reports, views and summaries to help users grab the progress of public voicing events in a comprehensive way. Analysis of event emotion orientation is a crucial part of public voicing system because the granularity and accuracy of emotional analysis are directly associated with the user's understanding of events. One of the main technologies relied on by public voice monitoring platform is the text classification technology.

Based on the public voicing system, this paper uses the BERT (Bidirectional Encoder Representation from Transformers) model to explore the multi-label text classification tasks of text for both short and long ones required by the public voicing platform. The experiment chooses news class text as its dataset, covering 14 long text categories and 15 short text categories. Considering that the long sample text exceeds the maximum length that the BERT model can handle, this paper designs five methods to explore the most suitable one for excessively long public voice text. They are head, tail, head and tail, text summary and splitting. The experimental results indicate that the head and tail should be regarded as the more appropriate one for the processing method of the dataset used in this experiment. Its performance is proved to be beyond what the current mainstream mode has achieved.

In addition, on the basis of the multi-label classification of text, this paper also studies the practical application of sensitive word detection in public voicing platform. Using DFA algorithm, this paper makes rapid and accurate detection of four different categories of sensitive words.

Eventually, in combination of the above-mentioned multi-label text classification model, this paper realizes some functional modules of public voice monitoring platform, and is capable of display and simulate these functions with public voicing platform.

Key words: public voicing system, multi-label text classification, BERT model, sensitive word detection, DFA algorithm

目 录

第一章 简介 1
1.1 课题研究背景及意义 1
1.2 国内外研究现状 1
1.3 研究目标 2
1.4 文章结构 3
第二章 相关技术研究 4
2.1 ELMo 4
2.2 OpenAI GPT 5
2.3 MT-DNN 6
2.4 XLNet 7
2.5 RoBERTa 9
2.6 SpanBERT 11
2.7 其他工作 11
2.8 敏感词检测技术研究 12
2.9 本章小结 12
第三章 算法设计与实现 13
3.1 Attention 13
3.2 Transformer 14
3.3 BERT 16
3.3.1 模型结构 16
3.3.2 输入输出 17
3.3.3 预训练 17
3.3.4 微调 18
3.4 实验：单标签多类分类 19
3.4.1 微调阶段使用的参数 19
3.4.2 数据预处理 20
3.4.3 训练流程 21
3.5 实验：多标签多类分类 23
3.6 敏感词检测技术研究与实现 24
3.6.1 DFA 算法 24
3.6.2 算法实现 24
3.7 本章小结 25
第四章 系统设计与开发 26
4.1 舆情系统介绍 26
4.2 功能模块设计 27
4.2.1 文本分类模块 27
4.2.2 敏感词检测模块 29
4.3 系统功能验证 30
4.3.1 上传文章并归档 30

4.3.2 按标签搜索文章/标题 30
4.3.3 新增标签分类 31
4.4 本章小结 32
第五章 实验结果 33
5.1 实验环境 33
5.2 短文本单标签多类分类实验 33
5.2.1 数据集数据分析 33
5.2.2 不同实验参数对比 33
5.2.3 不同模型结果对比 34
5.3 长文本单标签多类分类实验 34
5.3.1 数据集数据分析 34
5.3.2 不同处理方法对比 35
5.3.3 不同模型结果对比 36
5.4 多标签多类分类实验 36
5.4.1 数据集数据分析 36
5.4.2 实验结果 36
5.5 敏感词检测 37
5.6 本章小结 38
第六章 结论和未来工作展望 39
6.1 结论 39
6.2 未来工作展望 39
参考文献 41
谢辞 44

第一章 简介

1.1 课题研究背景及意义

随着因特网在全球范围内的飞速发展，网络环境成为了舆情主要来源。近些年网络舆情事件层出不穷。尤其随着移动互联网的快速发展，针对一定的社会空间内，民众对社会事件的发生、发展和变化持有不断变化的态度。为适应新形势下的网络安全工作需要，舆情系统应运而生。舆情系统可以对网络上各个平台的各类信息进行采集、分类、整合、筛选，实现对全网的舆情监测和舆情分析，从而使行业健康发展，维护社会稳定。

在舆情系统中，文本分类是一项基础而重要的技术。随着大数据时代的到来，数据的体量越来越大，内容也鱼龙混杂。如何从海量的数据中筛选出有价值的可以满足需求的特定数据，并使其发挥出本身的价值，成为了各行各业，尤其是互联网企业正在努力解决的关键问题。在这样的背景下，文本分类成为了解决上述问题的一项重要技术。

文本分类用电脑对文本集(或其他实体或物件)按照一定的分类体系或标准进行自动分类标记。它根据一个已经被标注的训练文档集合，找到文档特征和文档类别之间的关系模型，然后利用这种学习得到的关系模型对新的文档进行类别判断[1]。有效高效的文本分类方法,可以正确归纳文本主题，总结文章内容，充分利用数据的价值，进而开发出有针对性夫的个性化用户推荐系统,使其可以根据用户的个人兴趣来定位并推荐相关的文本资料，从而实现商业需求。这样的优势使得文本分类任务愈发受到各行各业，尤其是掌握大量数据的新闻媒体，电商等行业的青睐。在人工智能浪潮席卷全球的今天，文本分类技术已经日渐成熟，可以应用在情感分析，语义判断，上下文匹配等自然语言处理领域。

我们可以根据实际应用场景，将文本分类这一课题继续细分。根据类别的数目，可以分为二分类和多类分类；根据每条文本所属的类别个数，可分为单标签文本分类和多标签文本分类。本文主要研究的是基于 BERT 模型的单标签和多标签的文本分类，并把文本分类和敏感词检测功能结合起来，集成到舆情平台中发挥实际作用。

1.2 国内外研究现状

最初的文本分类仅仅通过分析本文内容，抓取文本关键词，通过堆和树等数据结构进行简单的分类操作(Maron, M.E. & Kuhns, J.L., 1960)[2]。比如基于规则的特征匹配，例如篮球映射到体育，电影映射到娱乐等。这样的方法虽然容易理解，但是存在诸多问题。首先，这个方法依赖专家系统，不同类别需要专门构建特征规则，费时费力。其次，运行效率低且准确率低，不能满足实际应用中对于文本分类准确率的要求。

近年来，随着人工智能的蓬勃兴起，文本分类这个课题越来越受到学者，从业人员和企业的关注。最初的解决方案是机器学习方法，如朴素贝叶斯分类(Pang B et al.,2002)[3]，决策树，支持向量机（SVM）等。基本原理是首先，专家们利用领域专业知识，创建合适的输入特征，然后把训练数据输入模型，通过算法拟合出最优的，即错误最小化的模型作为输出，最后把文本作为输入，寻找文本中的某些特征，根据这些特征通过训练好的模型将

文本分为不同的类别。这样的机器学习方法，在准确率上远高于人工分类的方法，但是仍有不足之处。首先，如何创建合适的输入特征，即特征工程[4]，是机器学习方法的最大瓶颈。特征工程这一步骤需要大量的专业知识以及不断地尝试，取得突破极为困难。其次，受限于计算能力，训练时使用的数据量有限，导致模型结构比较简单，存在欠拟合的问题。假如使用更大量的数据，又会出现过拟合的问题。

为了解决以上的问题，计算机科学家使用更多层的神经网络，使神经网络自动通过每一层产生适当的特征，从而避免了特征工程的复杂，创造出许多深度学习[5]方法如卷积神经网络（CNN），递归神经网络（RNN），长短期记忆模型（LSTM）(Zhou X., Wan X., Xiao J.,2011)[6]等。这些方法主要利用了文本，图像，语音等数据的连续，稠密的特点，利用其局部相关性，自动获取文本的特征。本文采用的 BERT（Bidirectional Encoder Representation from Transformers）[7]模型，就是基于深度学习方法，利用多层的双向 transformer 的网络结构实现了自动获取文本特征的能力。BERT 框架把原来的机器学习过程分成两个阶段：预训练（pre-training）阶段和微调（fine-tuning）阶段。首先，利用 Google 拥有的强大数据计算能力和海量的训练数据，训练出一个预训练模型，然后使用者可以根据自身的需求对预训练模型进行微调。这样既保证了模型具有极高的准确率，又对于使用者的计算能力没有太高的要求，从而在自然语言处理的许多任务中表现出色。而这种提供一个供其它任务迁移学习的模型，根据任务微调或者固定之后作为特征提取器的二阶段训练模式也成为深度学习的潮流。

自 2018 年 BERT 模型提出以来，BERT 模型被广泛应用在不同的自然语言处理任务中，均取得了出色的成绩。但是学者没有停止对 BERT 的研究，提出了许多针对不同自然语言处理任务的改进。其中比较出色的包括 MT-DNN[8]，XLNet[9]，RoBERTa[10]和 SpanBERT[11]等。

对于敏感词检测算法，目前主流的解决方案是 DFA 算法和 AC 自动机算法。而随着人工智能的发展，机器学习方法也被运用在敏感词检测领域。2000 年基于 Boosting 的机器学习方法被提出(Yu Hwan Kim et al.,2000)[12]，该方法利用朴素贝叶斯分类器来作为弱学习。2006 年，J.Polpinij[13]等人使用 SVM 支持向量机和贝叶斯算法分别对色情文本进行检测，并通过比较得出实验，结果证明 SVM 性能更优。2010 年， Malo[14]等人提出了基于 C4 .5 决策树策略的敏感内容检测算法。2013 年，出现了基于自适应主题建模的敏感信息文本内容检测新框架[15]，采用加权图获取敏感信息找出敏感话题。这些方法是都是通过机器学习识别文本潜在的语义，准确率无法达到百分之百，但是不需要建立敏感词字典树。这些机器学习方法的准确率仍待进一步的提升。

1.3 研究目标

本文的研究内容主要包括以下几点：

（1）基于 BERT 模型实现一个能够有效快速地将舆情系统中的舆情文本进行分类的模型。并针对原始 BERT 模型中存在的最大序列长度限制，设计多种长文本数据的预处理方法，探究最适合舆情系统的长文本处理解决方案。

（2）研究基于评价指标准确率（accuracy）的中文舆情内容长文本分类，基于 THUCNEWS 中文新闻长文本标注数据集的准确率不低于 95.35%。研究基于评价指标准确率（accuracy）的中文舆情内容短文本分类，基于今日头条中文新闻短文本分类数据集的准确率不低于 89.78%

（3）研究基于中文舆情文本关键词、语义的敏感内容检测技术。

（4）在现有舆情事件分析系统中集成上述方法。

本文就是使用预训练+微调的模式，把 BERT 模型应用在文本分类任务中，在实现 BERT 模型具有的高准确率的基础上，把模型运用到实际的应用中，使模型发挥自身的价值，并结合敏感词检测功能，实现一个完整的高效的业务流程。

1.4 文章结构

本文共包含六章，每章节大致内容概括如下：

第一章是简介，主要概述了文本分类和敏感词检测的研究背景及意义，分析当前主流的文本分类和敏感词检测技术的优缺点，并结合舆情系统的功能需求，明确本文的四个研究目标。

第二章是相关技术研究，包括文本分类和敏感词检测两个方面。文本分类方面主要介绍了深度学习方法中与文本分类技术相关的模型，敏感词检测方面介绍了目前主流的两种检测方法。

第三章是算法设计与实现，首先本文介绍了 BERT 模型的基本结构，运行原理和训练过程，然后介绍了本文在短文本和长文本两个数据集上基于 BERT 模型进行微调的过程，其中详细研究了本文在过长的文本上进行文本分类的五种文本预处理方法。最后实现了基于 DFA 算法的能够检测四种类别的敏感词检测功能。

第四章是系统设计与开发，对已有的舆情系统进行了简单阐述，主要介绍了本课题实现的功能模块（文本分类和敏感词检测），把第三章的文本分类算法和敏感词检测功能应用到真正的舆情系统中。

第五章是实验结果的对比分析，主要结合第三章提出的实验方法，分别分析短文本与长文本数据集上模型的表现，与其他模型结果进行对比，提出最适合舆情系统的文本多类分类解决方案。本文还探究了影响敏感词检测性能的因素，使得舆情系统更快速有效地完成一整套业务流程。

第六章是结论和未来工作展望，对本课题研究和实验存在的不足之处进行总结分析，提出未来工作可以进行改进的几个方向。

第二章 相关技术研究

如上文所述，随着人工智能的兴起，越来越多的文本分类方法被不断提出。本文主要介绍近年来占据主流的深度学习方法。在文本使用的 BERT[7]模型之前，ELMo 模型和 OpenAI GPT 模型是解决文本分类问题较好的方案，在 2018 年 9 月 BERT 模型提出以后，又出现了 MT-DNN，XLNet，RoBERTa, SpanBERT 等诸多针对 BERT 模型的改进方法。同时在敏感词检测领域，也存在 AC 自动机和 DFA 算法两种方式。本章将对这些相关技术进行分析和说明。

2.1 ELMo

2018 年 3 月份之前，在自然语言理解领域，词表征模型的结构是每个词对应一个向量，对于多义词无能为力。2018 年 3 月份提出的 ELMo(Embeddings from Language Models)[16]对此提出了一个较好的解决方案。不同于以往的一个词对应一个向量，预训练好的模型不再只是向量对应关系，而是一个训练好的模型。使用时，将一句话或一段话输入模型，模型会根据上下文来推断每个词对应的词向量。这样做之后明显的好处之一就是对于多义词，可以结合前后语境对多义词进行理解。比如 apple，可以根据前后文语境理解为公司或水果。作者认为好的词表征模型应该同时兼顾两个问题：一是词语用法在语义和语法上的复杂特点；二是随着语言环境的改变，这些用法也应该随之改变。

ELMo 是一种新型深度语境化词表征，可对词进行复杂特征(如句法和语义)和词在语言语境中的变化进行建模(即对多义词进行建模)。ELMo 使用的是一个双向的 LSTM 语言模型，由一个前向和一个后向语言模型构成，目标函数就是取这两个方向语言模型的最大似然（图 2）。众所周知，LSTM 的工作原理为：将单词转换成词嵌入，然后将上一时刻的输出/隐状态及第一步中的 word embedding 一并送入 LSTM，并得到输出及隐状态，最后将 LSTM 的输出与上下文矩阵相乘，得到一个列向量，再将该列向量经过 softmax 归一化。这就是一个基本的前向 LSTM 模型。后向语言模型，跟前向语言模型类似，给定后文来预测前文。ELMo 使用的双向 LSTM 语言模型，论文中简称 biLM，就是要最大化前向和后向的概率的似然概率的对数和（图 1）。

$$\sum_{k=1}^{N}\Big(\log p(t_k \mid t_1,\ldots,t_{k-1};\Theta_x,\overrightarrow{\Theta}_{LSTM},\Theta_s) + \log p(t_k \mid t_{k+1},\ldots,t_N;\Theta_x,\overleftarrow{\Theta}_{LSTM},\Theta_s)\Big)$$

图 1 ELMo 模型的目标函数

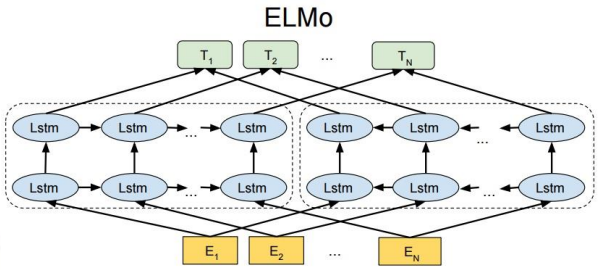

图 2 ELMo 的结构

对于每个单词，对于 L 层的双向 LSTM 语言模型，一共有 2L+1 个表征（前向 L 层的输出，后向 L 层的输出以及本身）。每层 LSTM 输出是不一样的，针对每个任务每层的向量重要性也不一样，所以对于每层向量，我们加一个权重，将每层的向量与权重相乘之后累

加，然后再乘以一个权重 γ。这样就得到了 ELMo 向量（图 3）。针对具体的自然语言理解任务，我们需要再次训练去得到的这个权重 γ。

$$\mathbf{ELMo}_k^{task} = E(R_k;\Theta^{task}) = \gamma^{task}\sum_{j=0}^{L} s_j^{task}\mathbf{h}_{k,j}^{LM}.$$

图 3 ELMo 向量的计算方式

所以对于 ELMo 的运行流程，我们可以这样归纳：1 产生预训练 biLM 模型。模型由两层 bi-LSTM 组成，之间用残差连接的方式连接起来。2 在任务语料上微调上一步得到的 biLM 模型。3 利用 ELMo 的词嵌入来对任务进行训练。把它们作为输入加到已有的模型中，可以明显的提高原模型的表现。这种预训练+微调的方式也被证明是有效的。而最终模型的表现也证明了 ELMo 和双向 LSTM 模型的成功：ELMo 能够同时区分语法和语义，能够在表示词语嵌入时考虑到上下文的信息，既消除了歧义，还能准确把握词性。

但是 ELMo 的局限性在于，LSTM 模型提取特征的能力不如 BERT 使用的 Transformer 结构，这影响了双向语言模型的表现。

2.2 OpenAI GPT

一个对文本有效的抽象方法可以减轻 NLP 对有监督学习的依赖。目前大多数深度学习方法依靠大量的人工标注信息，这限制了在很多领域的应用。此外，即使在可获得相当大的监督语料情况下，以无监督学习的方式学到的表示也可以提供显着的性能提升。

GPT[17]的核心思想就是先通过无标签的文本去训练生成语言模型，再根据具体的 NLP 任务（如文本蕴涵、QA、文本分类等），来通过有标签的数据对模型进行微调。在这篇论文中提出了半监督的方法，即结合了无监督的预训练和有监督的微调。首先，在未标记数据集上训练语言模型来学习神经网络模型的初始参数。随后，使用相应 NLP 任务中的有标签的数据地将这些参数微调，来适应当前任务。

GPT 也是预训练+微调的模式，只不过由 LSTM 结构进化为 Transformer 结构，提取文本特征的能力得到增强，但是 GPT 是一个单向的语言模型，没能同时把握上下文之间的联系，所以在自然语言理解任务中有局限性，但是单向的模型在自然语言生成领域比较有优势，因为符合人类生成语言的逻辑。

GPT 模型使用的是多层 Transformer 的 decoder 的语言模型，是由 12 层 transformer 模块组成的，使用最后的隐藏层来做不同的任务（图 4）。训练的两个阶段分为无监督的预训练（通过前 k-1 个词预测第 k 个词，从前往后，不断前进，属于单向的预测）和有监督的微调（在对模型预训练之后，采用有监督的目标任务对模型参数微调）。这与 BERT 模型的训练过程相似。

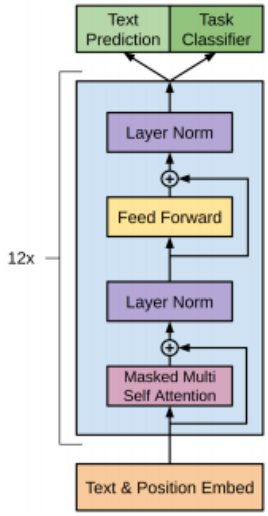

图 4 GPT 模型结构

对于有些任务，像文本分类，我们能够直接用上文的模型进行微调。另外的任务，比如问答系统，需要构造输入的句子对。又比如文本相似度任务，由于没有文本内部的先后顺序，可以链接前后两个文本作为输入等。

在 BERT 之后，GPT2 推出，使用 40GB 的高质量语料进行训练。同时针对文本输入的表示进行了改进，在生成类文本任务中有着很优秀的表现。另外,GPT2 提出了一种类似于“NLP 通用模型”的概念，作者认为语言模型=无监督多任务学习。相比于有监督的多任务学习，语言模型只是不需要显示地定义哪些字段是要预测的输出，所以，实际上有监督的输出只是语言模型序列中的一个子集。GPT2 使用的是带有任务信息的训练数据，以这种数据形式可以有监督地训练一个 single model。

2.3 MT-DNN

MT-DNN（Multi-Task Deep Neural Networks）[8]模型是一种多任务深度神经网络，用于跨多种自然语言理解任务的学习表示。多任务学习(MTL)的灵感来自于人的学习活动。在人类的学习生活里，我们往往会从之前的学习任务中获得知识来提高学习新任务的能力。多任务学习的优点有两个，一是弥补了有些任务的数据不足问题；二是得益于正则化效果，这种效果使得学习到的表示具有更强的泛化能力，防止模型过拟合。

MT-DNN 模型的思想是，多任务训练和预训练语言模型是互补的技术，可以结合起来提高文本特征表示的学习效果，进而提高各种自然语言处理任务性能的表现。因此，MT-DNN 模型的训练也分为两个阶段，即预训练和微调。但是，MT-DNN 在微调阶段用多个特定于任务的层进行多任务学习。主要有四种类型，其具体任务全部出自 GLUE（General Language Understanding Evaluation，通用语言理解评估）[18]：

（1）单句分类：CoLA[19]（Corpus of Linguistic Acceptability）任务是预测英语句子是否合乎语法，SST-2（Stanford Sentiment Treebank）任务预测电影评论是正向还是负向。

（2）文本相似度：这是一个回归任务。对于给定的句子对，模型计算二者之间的相似度。在 GLUE 中即为 STS-B[20]（Semantic Textual Similarity Bench-mark）这个任务。

（3）成对文本分类（文本蕴含）：对于给定的句子对，推理两个句子之间的关系。RTE[21]（Recognizing Textual Entailment）和 MNLI[22]（Multi-Genre Natural Language Inference）是语言推理任务，推理句子之间是否存在蕴含关系、矛盾的关系或者中立关系。QQP[23]（Q

uora Question Pairs）和 MRPC[24]（Microsoft Research Paraphrase Corpus）是判断两个句子的语义是否等价。

（4）相关性排序：给定一个问题和一系列候选答案，模型根据问题对所有候选答案进行排序。QNLI[25]（Stanford Question Answering）任务是预测候选答案中是否包含对问题的正确答案。这个任务重排了候选答案，将正确答案排在更前。

MT-DNN 的体系结构主要可以概括为：底层（即 BERT 的预训练阶段）在所有任务中共享，顶层（即上述四种类型的任务）代表特定任务的输出（图 5）。输入是一个句子或一对组合在一起的句子对。训练模型时把所有数据合并在一起，每个 batch 只有单一任务的数据，同时会带有一个 task-type 的标志，这样模型就可以分辨不同的任务，进行不同的 loss 计算。相比于交替训练（先训练任务 A 再训练任务 B），这个训练方法可以有效避免偏向某个任务，近似地优化所有多任务的和。

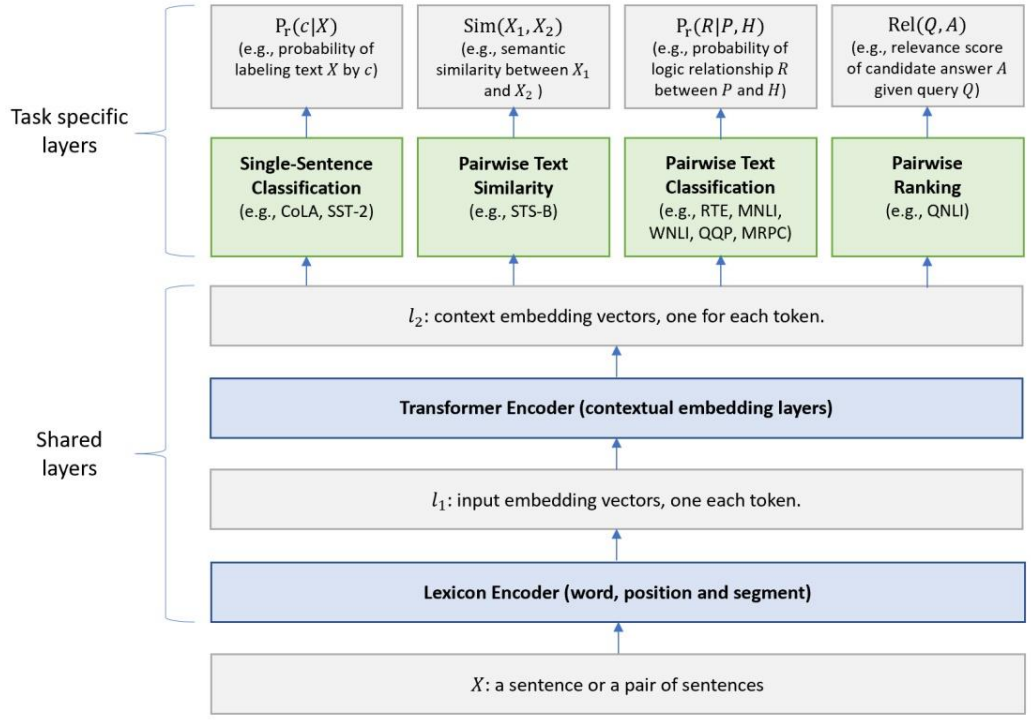

图 5 MT-DNN 模型结构[8]

从训练结果上来看，MT-DNN 模型在 GLUE 中除了 WNLI 之外的全部任务都超越了已有的模型，尤其是在数据集较少的时候，MT-DNN 相较于 BERT 有着极大的提升。这样的结果可以证明，预训练模型+多任务学习的方式，确实可以提升了模型的表现，而且可以有效解决数据集不足的问题，使得模型具有更好的泛化能力。但是 MT-DNN 模型也有有待提高的地方，比如作者提到的更有效的训练方法，更深度的权重共享[26]等。

2.4 XLNet

XLNet[9]也是学者对 BERT 模型的一种改进。XLNet 首先提出了自回归语言模型（Autoregressive LM）以及自编码语言模型（Autoencoder LM）的分模型分类方法。自回归语言模型是指根据前文内容预测接下来将要出现的单词，换言之就是自左向右（或从右向左）的语言模型，自回归语言模型的优势是擅长生成式自然语言处理任务。 因为在生成上下文时，通常是单向的，自回归语言模型很自然地适用于此类 NLP 任务。但是，它只能使用前

向上下文或后向上下文，这意味着它不能同时使用前向和后向上下文。这样的缺点使得预测的准确率仍有提高的空间。而自编码语言模型可以同时关注上下文，因为自编码语言模型的目的是从损坏的输入重建原始数据，方法是在输入中随机用伪码标记代替一部分单词，然后预根据上下文单词来预测这些被替换掉的单词，从而得到原句。BERT 模型就是一个很典型的自编码语言模型，因为 BERT 模型预训练时的任务之一就是 Masked Language Model。这样的训练方法可以兼顾前文和后文，能比较自然地融入双向语言模型，相较于自回归语言模型是一个进步。但是自编码语言模型也有缺点。此训练方法在输入侧引入伪码标记，但是 Fine-tuning 阶段是看不到伪码标记的，导致预训练阶段和微调阶段不一致的问题，从而引入了一些人为误差。而且该方法假设被伪码代替的词之间彼此独立。而在实际数据中，被代替的词之间可能会存在相关性，我们想要语言模型学习这种相关性来对这些被代替的词进行更准确的预测[27]。

XLNet 试图找到一种方法，融合自回归和自编码两种语言模型，集合两者的优点，避免两者的缺陷。XLNet 采用的是自回归语言模型，为了解决上述的缺点，作者提出了排列语言模型（Permutation Language Modeling）。原始的自回归语言模型是把序列的值按顺序进行建模，所以是单向的。XLNet 采用的方法是最大化所有可能的序列的因式分解顺序的期望对数似然，即随机排列组合句子中的单词，在排列组合后的所有可能出现的输入序列里，再抽取一部分输入序列作为预训练的输入。假如我们有一个序列 abcd（对应图 6 中 1234），如果我们要预测字母 c，先对该序列进行因式分解，最终会有 24 种排列方式，假如抽取 cbda 的排列方式，因为 c 的左边没有其他的值，所以该情况无需做对应的计算；假如抽取 bdca 的排列方式，模型就会保留 b，d 的信息，从而使得上下文信息都能被保留，解决了传统自回归语言模型的缺点。而且这种方法避免了采用伪码标记，也解决了 BERT 模型的缺点。

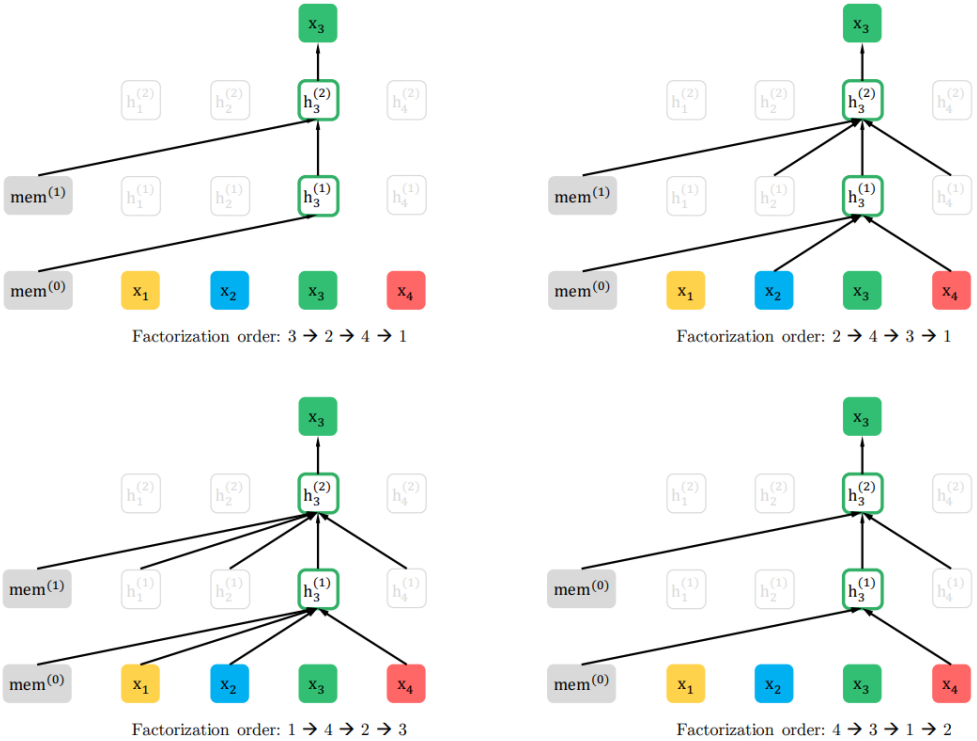

图 6 排列语言模型的一个实例[9]

但是这种方法在实现的过程中也有困难。虽然排列语言模型能满足目前的目标，但是对于普通的 transformer 结构来说存在问题。传统的 Transformer 结构中，某个单词的内容和位置向量在输入到模型前就已经加在一起了，后续的隐向量同时具有内容和位置的信息。但是，

我们希望在预测下一个词时只能提供位置信息，不能提供内容相关的信息。因此模型需要同时做两件事，首先它需要预测自己到底是哪个字符，其次还要能预测后面的字符是哪个。为了解决这个问题，论文中提出来新的机制，来实现目标位置感知——双流自注意力机制（Two-Stream Self-Attention）。该机制包括两个部分，第一是内容流自注意力（图 8），即标准的 Transformer 单元的计算流程，同时编码了上下文和自身的内容和位置信息；二是查询流自注意力（图 7），包含上下文的内容信息和目标的位置信息，但是不包括目标的内容信息。两个流共同计算，进行模型的训练。

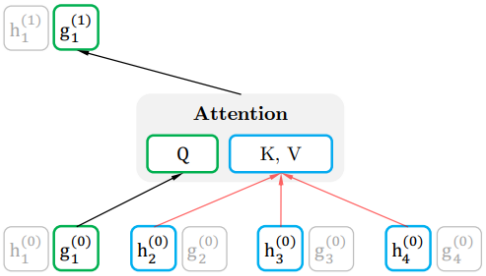

图 7 双流注意力机制：查询流[9]

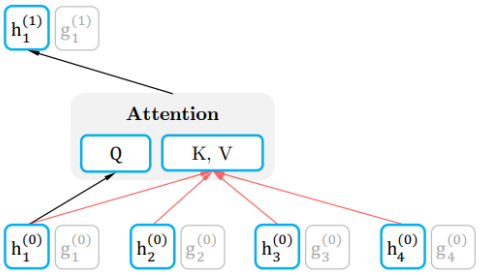

图 8 双流注意力机制：内容流[9]

同时，XLNet 模型还集成了 transformer-xl 的两个最重要的技术点，即片段循环机制和相对位置编码。片段循环机制主要是为了解决超长序列的依赖问题，首先取第一个段进行计算，然后把得到的结果的隐藏层的值进行缓存，第二个段计算的过程中，把缓存的值拼接起来再进行计算。该机制不但能保留长依赖关系还能加快训练，因为每一个前置片段不需要再重新计算。但是 BERT 模型中采用的绝对位置编码，没法区分之前存储的信息到底是哪一个片段里的，这就导致了一些位置信息的损失。因此 XLNet 采用相对位置编码，在计算 attention 的时候根据当前的位置和要参考的位置的相对距离来"实时"体现在 attention 的计算中。

在实验中，XLNet 的表现也相较 BERT 有明显提升，尤其是在较长的文本上的表现。排列语言模型的机制对于自回归和自编码两种模式的融合，片段循环机制的引入，更大的数据规模，这些因素共同提升了 XLNet 在自然语言处理领域的表现。

2.5 RoBERTa

RoBERTa（a Robustly Optimized BERT Pretraining Approach）[10]是 BERT 的改进版。与 BERT 使用相同的网络结构，但是从以下四个方面对 BERT 进行了提升：

首先是更大的数据规模，更大的 batch size，更长的训练时间。RoBERTa 在训练过程中使用了更大的 bacth size。尝试过从 256 到 8000 不等的 bacth size。训练数据包括 160GB 纯文本，相较于最初的 BERT 使用 16GB 的数据集和英语维基百科进行训练，RoBERTa 显然会具有更好的性能。

第二，RoBERTa 使用动态掩码（dynamic masking）的方式进行预训练。原始 Bert 模型对每一个输入序列随机选择 15%的单词替换成掩码，为了消除与下游任务的不匹配，还对这 15%的单词进行分类：

（1）80%替换成掩码；

（2）10%不变；

（3）10%替换成其他词。

但整个预训练过程，这 15%的单词一旦被选择就不再改变，每个 epoch 都是重复，后续的每个训练步都采用相同的掩码。这就叫做静态掩码。作者为了研究掩码对于模型性能的影响，设计了两种改进方式进行了对照。第一种方式是在预处理的时候将数据集用不同的掩码

拷贝 10 次，等价于原始的数据集采用 10 种静态 mask 来训练。(即图 9 所示的 static 方法）第二种方式是不在预处理的时候执行掩码标记，而是在每次向模型提供输入时动态生成掩码。经过实验发现，第一种方法与原始 BERT 的静态掩码方式效果相同，而第二种动态掩码的方式可以提升模型的性能。所以在 RoBERTa 中，作者采用动态掩码的方式进行预训练。

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

图 9 各种掩码方式的准确率对比[10]

第三，RoBERTa 研究了原始 BERT 预训练过程中，NSP（Next Sentence Prediction，下一句预测）任务的效果。NSP 任务的目的是原为了捕捉句子之间的关系，方法是输入连续的多个句子组成的文本 A 和 B，判断这两组句子是否是连续的。在训练集的数据中，50%的 B 是 A 的下一组句子，50%的 B 是随机抽取的。近来有学者对于 NSP 任务提出质疑，认为这个任务不是必要的。RoBERTa 的作者对此设计了不同的训练方法进行了实验：

（1）原始 BERT 的 NSP 任务

（2）用单个句子代替原来的多个句子，且句子长度不超过 512。 由于这些输入明显少于 512 个单词，因此增加 batch size 的大小，以使单词总数保持与原始 BERT 的 NSP 任务相似。

（3）舍弃 NSP 任务，直接用来自同一个文档或者不同文档的连续多个句子作为输入，单词总数不超过 512 。一次输入可能会跨越文档边界，如果跨文档，则在上一个文档末尾添加文档边界标记。（

（4）舍弃 NSP 任务，直接用来自同一个文档的连续多个句子作为输入，单词总数不超过 512 。

由于在文档末尾附近采样的输入长度短于 512 个单词，因此在此情况下动态增加 batch size 大小以达到与方法（3）相似的单词总数。经过实验，方法（4）的性能表现最好。但是由于方法（4）需要动态调整 batch size，为了避免不必要的麻烦。RoBERTa 的作者采用了实验结果最接近方法(4)的方法（3）。这两种方法的表现都明显优于原始 BERT 的方法（如图 10）。

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

图 10 4 种方法的准确率对比[10]

第四，RoBERTa 采用了 byte-level 的字节对编码（BPE）的方式处理自然语言语料库中

常见的大量词汇。字节对编码不依赖于完整的单词，而是依赖于子词(sub-word)单元，这些子词单元是通过对训练语料库进行统计分析而提取的。原始 BERT 使用的是 character-level 的大小为 30K 的字节对编码，而 RoBERTa 使用 bytes 而不是 unicode 字符作为子词的基本单位，可以编码任何输入文本而不会引入未知标记。词表大小也从 30K 提升到 50K。这样的改进为原始 BERT 模型的预训练增加了 1500 万以上的参数，有效的提高了模型的表现。

RoBERTa 设计实验评估了许多设计训练时的策略，并通过更高级的测试环境实现了对原始 BERT 模型的改进，而且在测试集上的表现甚至超越了 XLNet，证明了设计训练策略的重要性。

2.6 SpanBERT

SpanBERT[11]也是针对 BERT 在预训练阶段的训练策略做出的改进。与 RoBERTa 相似的是，SpanBERT 在训练时也舍弃了 NSP 任务，SpanBERT 的作者给出了如下的解释：（1）相比起两句拼接，一句长句，模型可以获得更长上下文（类似 XLNet 的一部分效果）（2）在 NSP 的负例情况下，基于另一个文档的句子来预测词，会给 Masked Language Modeling 任务带来很大噪音。

SpanBERT 的创新之处在于：

第一，提出了更好的掩码方案（图 11）。对于原始 BERT 模型，训练时会随机选取整句中的最小输入单元来进行遮盖。这样会让本来应该有强相关的一些连在一起的字词，在训练时是割裂开来的。这样会导致训练结果的表现下降。为了解决这个问题，有学者尝试将这样具有较强相关性的单词连接起来一起用掩码标记（BERT WWM 模型）。还有学者更进一步，直接将由几个词组成的实体都遮盖掉（如 ERNIE 模型）。SpanBERT 的做法是，根据几何分布，先随机选择一段（span）的长度，之后再根据均匀分布随机选择这一段的起始位置，最后按照长度遮盖。经过作者的实验，这种 random span 的方法是优于以上两种改进方法的。

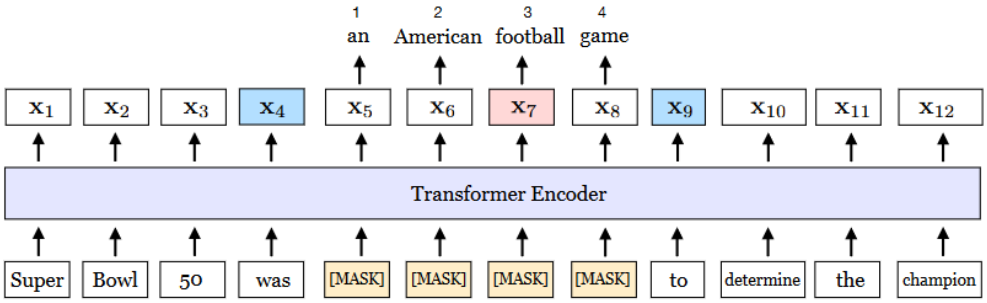

图 11 SpanBERT 掩码方法[11]

第二，SpanBERT 在训练阶段新加入了一项任务：Span Boundary Objective。在训练时取 Span 前后边界但不在 Span 内的两个词，然后用这两个词向量加上 Span 中被遮盖掉词的位置向量，来预测原词。这项新任务的加入，大大提高了 random span 的掩码方法的表现。

在以上创新点的基础上，作者主要对抽取式问答，指代消解，关系抽取，还有 GLUE 做了实验。发现 SpanBERT 的表现普遍强于 BERT，而且 SpanBERT 尤其在抽取式问答上表现好，这与它新加入的 SBO 预训练目标关系较大。

2.7 其他工作

除了以上提到的已有模型，还有学者对于 BERT 模型在不同任务下的表现进行了研究。

比较有代表性的如：

（1）BERT 模型在 Argument Reasoning Comprehension 任务中的表现[28]。ARC 任务是在给定的前提下，对于某个陈述，对含义相反的两个依据进行判断，判断哪个可以支持之前提到的陈述。作者进行了实验，得出了结论并提出了合理疑问，认为 BERT 模型并没有真正学习到做出推测所需要的语义信息，只是利用了某些出现频率很高而且对于推测有很大影响的词，比如 not，are 等。作者又设计实验，创建了对抗数据集（Adversarial Dataset），重新进行实验，发现 BERT 模型在 ARC 任务上的表现并没有达到预期。

（2）BERT 模型在 Natural Language Inference（自然语言推理，NLI） 任务中的表现[29]。作者首先假设在 NLI 中表现好的模型可能利用了三种启发式的特征，在任务的前提中就给了模型一些提示，有如下三种：语义重叠，对应的推理是前提的子序列或者对应的推理是前提的子串。在这个假设的基础上，作者进行实验并发现，这样的特征在 MNLI 训练集中的许多数据点中都存在，而且对应选项为不正确的数量远少于选项为正确的数量。为了减轻这种情况，作者构造了 HANS 数据集，使得两种类型样本能够均匀分布，并且标记了每条数据的前提是否涉及上述几种特征。进行实验时，新的模型在 MNLI 数据集上进行微调，在 HANS 数据集评测，结果发现，选项为正确的数据点中模型的表现优秀，但在推理错误的数据点中模型表现达不到预期。这一实验结果证明，BERT 模型在 NLI 任务中的确过度利用了某些具有启发式特征的信息。最后，作者在原始的训练集中加入了一些的 HANS 数据集中的数据，构造了新的 MNL + 数据集，让 BERT 模型在新数据集上进行微调，最终获得了更优秀的结果。

虽然 BERT 在许多方面仍有提高的空间，但 BERT 作为这些新模型的基础，仍具有可观的研究价值和提升潜力。接下来我们将通过 BERT 方法，实现在真实平台上可用，有效，高效的文本多分类模型，并研究 BERT 模型在文本分类任务中的表现。

2.8 敏感词检测技术研究

最普通的敏感词检测，就是用遍历的方法，从敏感词库中逐条读取敏感词，在待检测文本中进行搜索匹配。实现起来非常容易，但是问题很大：当待检测文本比较长，敏感词汇比较多的时候，这种方法效率很低，严重影响用户体验。

现在被广泛应用的敏感词检测方法有两种。第一种是本文采用的敏感词检测方法：DFA 算法，即确定有限状态机，将字符比较转化为了状态转移。第二种方法是 Aho-Corasick 自动机算法，简称 AC 自动机[30]。该算法是 DFA 的一种实现。这两个算法有两个特点，一个是扫描文本时，只需要一次遍历就可以全部检测；另一个是时间复杂度是固定的，为 O(n)，与关键字的数目和长度等因素没有关系，因此具有很高的效率，不影响用户的使用体验。

2.9 本章小结

本章主要介绍了文本分类和敏感词检测相关技术。详细阐述了文本分类任务中主流的表现优秀的模型及其优缺点，为接下来针对 BERT 模型作出有针对性的改进打下基础。同时，介绍了两种不同的敏感词检测方法，为舆情系统的业务流程的实现提供了解决方案。

第三章　算法设计与实现

文本分类的方法的种类已经在第一章介绍过，有基于规则的特征匹配，还有朴素贝叶斯分类，决策树，支持向量机等传统机器学习方法，还有本文采用的基于深度学习的方法。这些算法的实现和最终效果也不尽相同，有每条文本只能有一个类别的单标签分类，也有每条文本可以属于多个类别的多分类；有判断是否属于某项类别的二分类，也有在很多在类别中进行判断的多类分类。本章节将重点针对舆情系统中新闻类短文本和长文本进行单标签多类分类，解读 BERT 模型的工作原理并针对舆情系统对 BERT 模型的局限之处进行优化与测试，同时研究 BERT 模型在多标签分类中的应用。

文本分类是通过提取文本信息，推测文本主题所属的类别的过程。文本分类的第一步，就是把文本信息通过恰当的方式进行表示。当前主流的文本表示方法，往往只能够单向地读取文本信息，这意味着模型只能从文本的上文或者下文获得信息，推测接下来的文本。这样的表示方法是不够优秀的。我们希望可以在同一个模型中，同时获得被预测词上文和下文的信息，提升预测的准确率。同时， BERT 模型在最大能够处理的文本长度上有 512 字的限制，在新闻类长文本这种可能篇幅较长的文本上无法直接使用。

针对上述存在的问题，本文基于原始 BERT[7]模型实现文本分类任务。BERT 的全称是 Bidirectional Encoder Representation from Transformers，即 Transformer[31]结构中双向的 Encoder。BERT 模型使用三种输入相加的文本处理方式来表示文本的内容信息和位置信息，实现同时获取上下文信息的功能。同时，本文在长文本预处理过程中尝试了不同的方法，尝试结合截断，拆分，文本摘要等方式优化长文本分类的结果。

3.1 Attention

要介绍 Transformer 结构，首先要介绍注意力 Attention[32]这个机制。注意力的灵感来自于人处理信息时采取的一种手段，具体表现为我们如何对信息的不同区域关注度会有所区别，通常对于感兴趣或有更大价值的部分往往会分配大量的注意力。在深度学习中，注意力可以大致理解为对于某一个向量关注的程度如何，我们使用注意力向量来估计关注的部分和其他元素之间的关系强弱，并将不同部分的值的和用注意力向量加权得到的结果作为目标的近似值。

在自然语言理解领域内，注意力这一机制大放异彩。最初的自然语言理解任务中，解决方案大多为 Sequence To Sequence 模型。目的是将一个输入序列转换为一个新的目标序列，并且输入序列和目标序列的长度可以是不固定的。为了实现这一目的，模型主要做两件事情：

（1）将输入序列数据的信息压缩到一个固定长度的上下文向量（context vector）中，得到的表示向量可以较好的包含整个输入的信息。这个阶段也叫做 Encoder。

（2）使用上下文向量对模型进行初始化，然后输出转换后的向量。这个阶段也叫做 Decoder。

这种传统的模型可以完成机器翻译、生成问答对话、句法分析等自然语言处理任务，但这个机制有一个关键的不足：使用固定距离的上下文向量，会导致其不能记住较长的句

子。一旦完成了对于某个输入的处理，它就会忘记之前已经学到的部分。而且，上下文向量可能无法完全表示整个序列的信息。

为了解决这个问题，科学家创造出 Attention 机制。带有 Attention 机制的 Encoder-Decoder 模型中，上下文编码就不在是输入序列的直接编码，而是要从序列中获取每一个元素的重要程度，然后按元素的重要程度进行合并。在 Encoder[33]将输入的序列元素进行编码时，得到的不在是一个固定的上下文编码，而是存在多个隐藏状态，且最后的上下文编码由不同的隐藏状态以不同的权重参数组合而成。所以通过该机制，模型能够选择性地关注输入序列的有用部分，从而学习它们之间的“对齐”，即将原文的片段与其对应的译文片段进行匹配（如图 12 所示）。

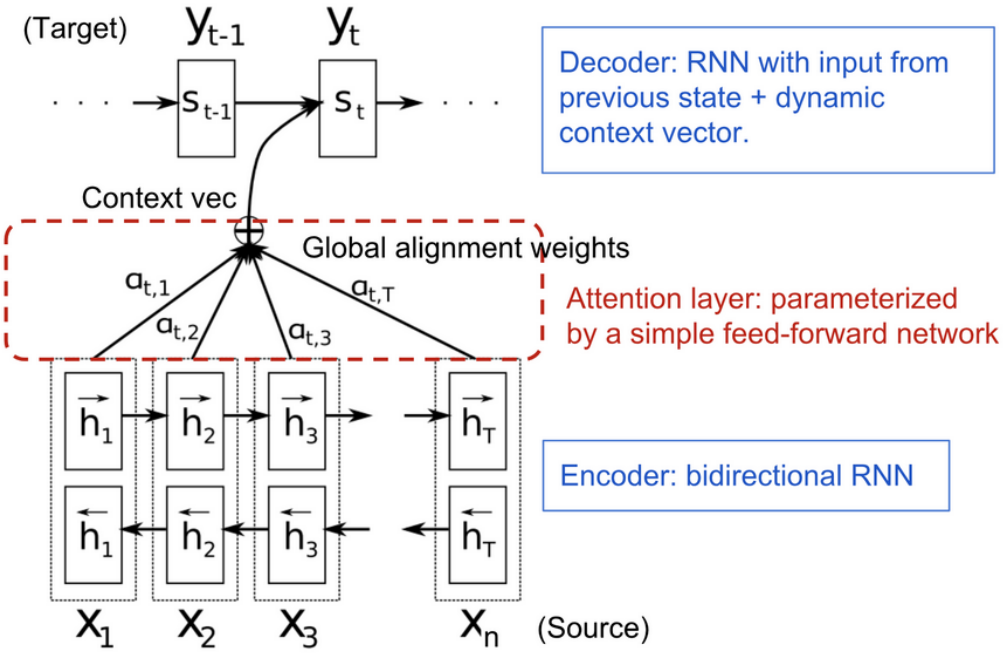

图 12 Attention 机制

在此基础上，又根据不同的输出计算方式，分为 Hard Attention 和 Soft Attention。Soft Attention 中又根据加权求和的集合不同，分为 Global Attention 和 Local Attention。

Attention 机制使模型能够有效捕捉文本的全局联系和局部联系，模型复杂度小，参数少，而且可以通过并行计算减少模型训练时间。Attention 机制作为一种思想，可以根据实际情况和多种模型进行结合。Google 就在 Attention 机制的基础上，设计了 Transformer 这样一种结构，并以 Transformer 为基本单元，设计出 BERT 模型，在自然语言理解领域取得了非凡的成就。

3.2 Transformer

我们首先介绍 Self-Attention 的机制。Self-Attention 是 Attention[32]的特殊形式。为了研究一个单词在单词所在的句子中受其他单词影响的情况，把要处理的单词作为输入，把单词所在原句的各个部分通过 Encoder 得到对应的隐藏状态，按照上一部分介绍的计算方法进行计算。Self-attention 使我们能够学习到当前单词和句子中前一部分之间的相关性。

在 Self-Attention 的基础上，我们使用三种不同的线性变换对输入和隐藏状态分别进行投影，利用这些投影分别计算注意力，最后将三个计算结果拼接起来，这种方式被称作

Multi-head Attention（如图 13）。Multi-head Attention 使得模型能够在不同的位置共同关注来自不同表示子空间的信息，如果只有一个 Attention，平均就会抑制这种情况。

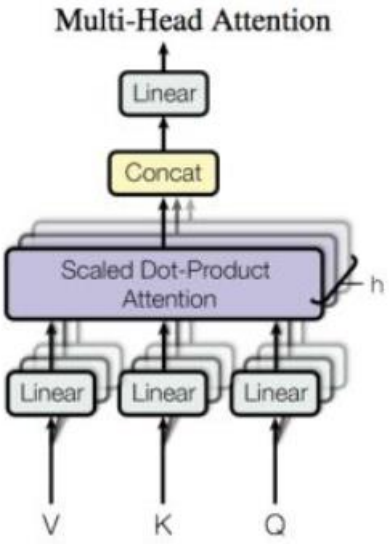

图 13 Multi-head Attention

下面介绍 Transformer 结构。Encoder 部分用于生成基于注意力的表示。首先，Encoder 部分把输入进行位置编码，这是为了明确关于单词在句子中位置的信息。然后将进行了位置编码后的数据与 embedding 数据求和，这样把相对位置信息加入到输入数据中。将得到的词向量进行 Multi-head Self-Attention 计算，然后进行残差连接（Residual Connection，通过将前一层的信息传递到下一层，解决多层神经网络训练困难的问题）和层归一化（Layer Normalization，通过对层的激活值的归一化，加速模型的训练过程，使其更快的收敛）。再把得到的结果输入 position-wise 类型的全连接前馈网络层。然后再经过残差连接和层归一化得到输出。Encoder 默认有 6 层。

Decoder 默认也是 6 层，每次计算之后也都会进行残差连接和层归一化。这一部分的输入有两个，一个是上一级 decoder 的输出，一个是 Encoder 阶段的输出。对于上一级 Decoder 的输出，首先进行一次 Masked Self-Attention，其中 Mask 掉的是当前时刻以后的信息，这样可以保证预测时不会利用被预测单词之后的文本信息。然后把得到的输出作为下一环节的输入，与 Encoder 部分的输出一起进行 Attention 的计算（注意这一次不是 Self-Attention，而是普通的 Attention 计算）。最后把得到的结果输入全连接层得到输出。decoder 的输出会经过线性层和 softmax 层得到最终的结果。

最终 transformer 的结构如图 14 所示。

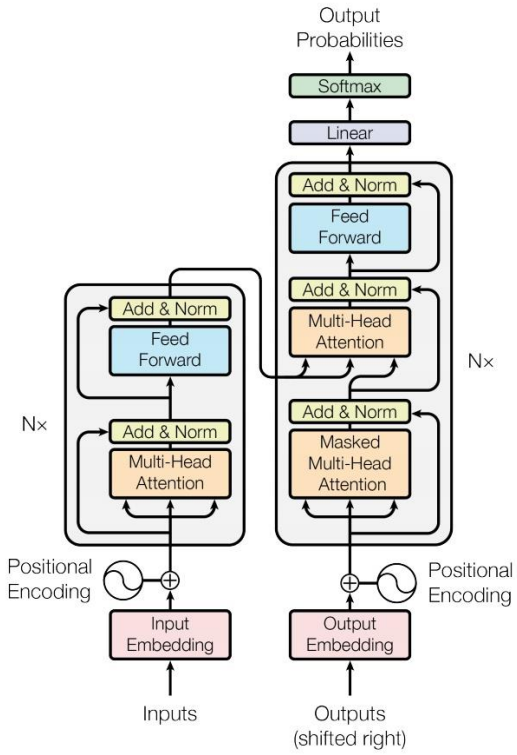

图 14 Transformer 的结构

3.3 BERT

2018 年 9 月 Google 提出了 BERT[7]模型，在大多数自然语言处理任务中刷新了最好成绩，并且准确率提升明显。我们将从模型结构和训练方法两方面介绍 BERT 模型并分析它取得如此成绩的原因。

3.3.1 模型结构

BERT 就是以 Transformer[31]结构为基本元素，通过预训练和微调的方法训练出的模型。BERT 模型最大的特色在于，通过对处于所有层中的上下文联合作用，来对深度双向表示的未标记文本进行预先训练。在 BERT 之前，有两个策略将预训练的语言表征应用到下游任务：基于特性（feature-based)和微调。前者（例如 ELMo）采用将预训练表征作为额外特征的具体任务架构。后者（例如 OpenAI GPT）引入了最少的，可以根据任务不同进行调整的参数，可以通过简单的微调过程对下游任务进行训练。这两种方法在预训练中使用相同的目标函数，使用单向语言模型来学习通用语言。BERT 的作者的观点是，单向语言模型限制了预训练阶段获取文本表征的能力。这将会限制对预训练中使用架构的选择。这种限制对句子级别的任务而言影响不大，但对于一些必须双向合并语境，像问答这样的单词级别的任务时这种限制就变得影响很大。为了减轻这种单向模式的限制，BERT 采用 Masked Language Model 作为预训练目标并且采用深层的双向 Transformer 组件（使用双向 self-attention，使得每一个单词可以联系到其之前以及之后的单词），结构如图 15 所示。

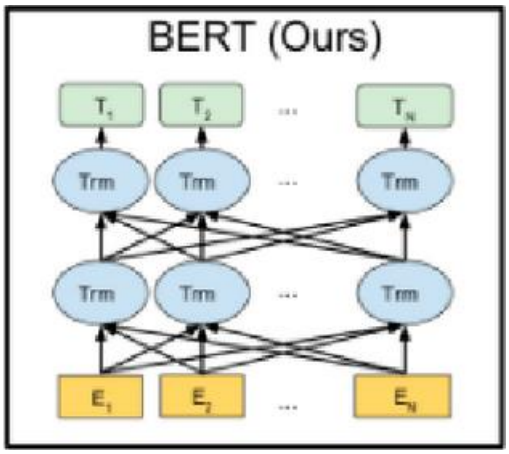

图 15 BERT 模型的结构

BERT 模型还根据模型大小分成两种：BERTBASE (L=12, H=768, A=12, Total Parameters=110M) and BERTLARGE (L=24, H=1024,A=16, Total Parameters=340M)。其中 L 表示层数，H 表示隐藏大小，A 表示 self-attention heads 的数量。

3.3.2 输入输出

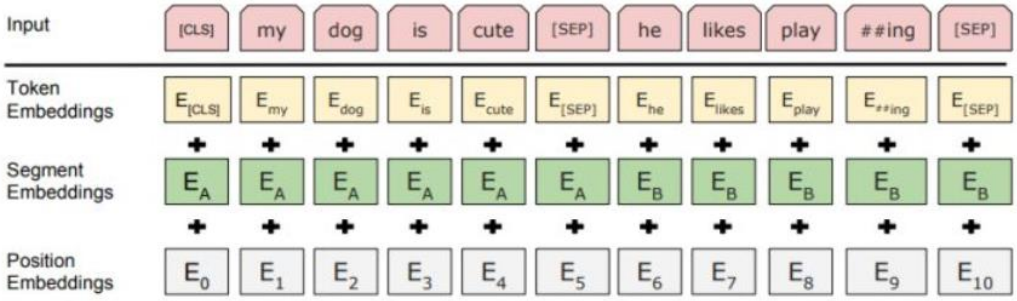

图 16 BERT 模型的输入[7]

由于 BERT 是一个预训练模型，其必须要适应各种各样的自然语言任务，因此模型所输入的序列必须有能力包含一句话（文本情感分类，序列标注任务）或者两句话以上（文本摘要，自然语言推断，问答任务）。所以输入文本的嵌入由三种嵌入求和而成（如图 16 所示）：

（1）Token Embeddings 是词向量，第一个单词是 CLS 标志，可以用于之后的分类任务。

（2）Segment Embeddings 用来区别两种句子，因为预训练要做以两个句子为输入的分类任务。

（3）Position Embeddings 将单词的位置信息编码成特征向量。其中，为了能在单词的有效性和字符的灵活性之间取得一个折中的平衡，作者还使用了 WordPiece[34]算法，将单词划分成一组有限的公共子词单元。

3.3.3 预训练

（1）任务一 Masked Language Model[35]

标准条件语言模型只能从左到右或从右到左进行训练，但是双向的模型结构允许每个单词间接"看到自己"，并且模型可以在多层语境中预测目标单词。

为了训练一个深度双向表示，BERT 在预训练时随机屏蔽了一部分的输入单词，然后预

测那些被屏蔽的单词。此过程即为" Masked Language Model "（MLM），在文献中通常被称为完形填空任务。在这种情况下，与掩码对应的最终隐藏向量会像标准 LM 中一样（输入到全连接，然后用 softmax 输出每个 token 的概率，最后用交叉熵计算 loss）。在所有实验中，我们随机屏蔽每个序列中 15%的经 WordPiece 嵌入后的单词。

虽然这样的预训练任务可以获得一个双向预训练模型，但缺点是，这项任务没有正确匹配预训练和微调这两个过程，因为掩码这个单词在微调期间不会出现。为了改进这一缺点，BERT 并不总是用固定的掩码标记去替换那些被遮蔽的单词。输入样例生成时会随机选择 15% 的单词进行替换并预测。如果第 i 个单词被选中，有 80%的可能将其替换为掩码标记，10%的可能替换为随机的单词，10%的可能保持原单词不变。随后，第 i 个单词将和交叉熵损失一起被用来预测原始单词。

（2）任务二 Next Sentence Prediction (NSP)

许多下游任务都是理解两个句子之间的关系，而语言模型无法直接捕获这些关系。为了培养模型了解句子关系的能力，BERT 在预训练过程中进行了语句对预测的任务。具体过程为，每个预训练示例都会选择两个句子 A 和 B ，B 有 50% 的可能是 A 的真正的下一个句子的可能性为 50%（标记为 IsNext），还有 50%的可能，B 是来自语料库的随机句子（标记为 NotNext）。尽管这项任务很简单，但这项任务对问答和自然语言推理都非常有用。因为在 BERT 模型之前，只有句子级别的嵌入被转移到下游任务，但是 BERT 模型把包括句子和单词等所有参数都进行了转化，用来初始化最终任务模型参数。

（3）BERT 使用的预训练数据

这两个任务所需的数据其实都可以从无标签的文本数据中构建，样例为：

Input1=[CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]

Label1=IsNext

Input2=[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]

Label2=NotNext

把每一个训练样例输入到 BERT 中可以相应获得两个任务对应的 loss，再把这两个 loss 加在一起就是整体的预训练 loss。（也就是两个任务同时进行训练）

对于预训练语料库，BERT 使用 Books 语料库[36]（800M 单词）和英语维基百科（2500M 单词）。BERT 模型只提取维基百科的文本段落，忽略列表、表格和标题等其他文本。并且十分重要的一点是 BERT 使用文档级语料库而不是杂乱的句子级语料库（如 Billion Word Benchmark[37]）来提取冗长的连续序列。这有利于模型具备抽象连续长序列特征的能力，更好地学习句子之间的关系。

3.3.4 微调

经过预训练的 BERT 模型可以通过一个额外的输出层进行微调，从而创造出适用于更多下游任务的模型，而不需要针对具体的任务进行大量架构上的修改。部分任务的微调方式如图 17 所示。transformer 中的 self-attention 机制允许 BERT 通过交换适当的输入和输出来建模许多下游任务（无论是涉及单个文本还是文本对），这使得微调非常简单。例如：

（1）在分类任务中，例如情感分析等，只需要在 Transformer 的输出之上加一个分类层。

（2）在问答任务（例如 SQUAD v1.1[38]）中，问答系统需要接收有关文本序列的问题，并且需要对答案进行标记。 可以使用 BERT 学习两个标记答案的开始和结尾的向量来训练问答模型。

（3）在命名实体识别（NER）任务中，模型需要获取文本序列，对文本中的各种类型的实体进行标记。我们可以用 BERT 模型将每个单词的输出向量送到预测 NER 标签的分类层。

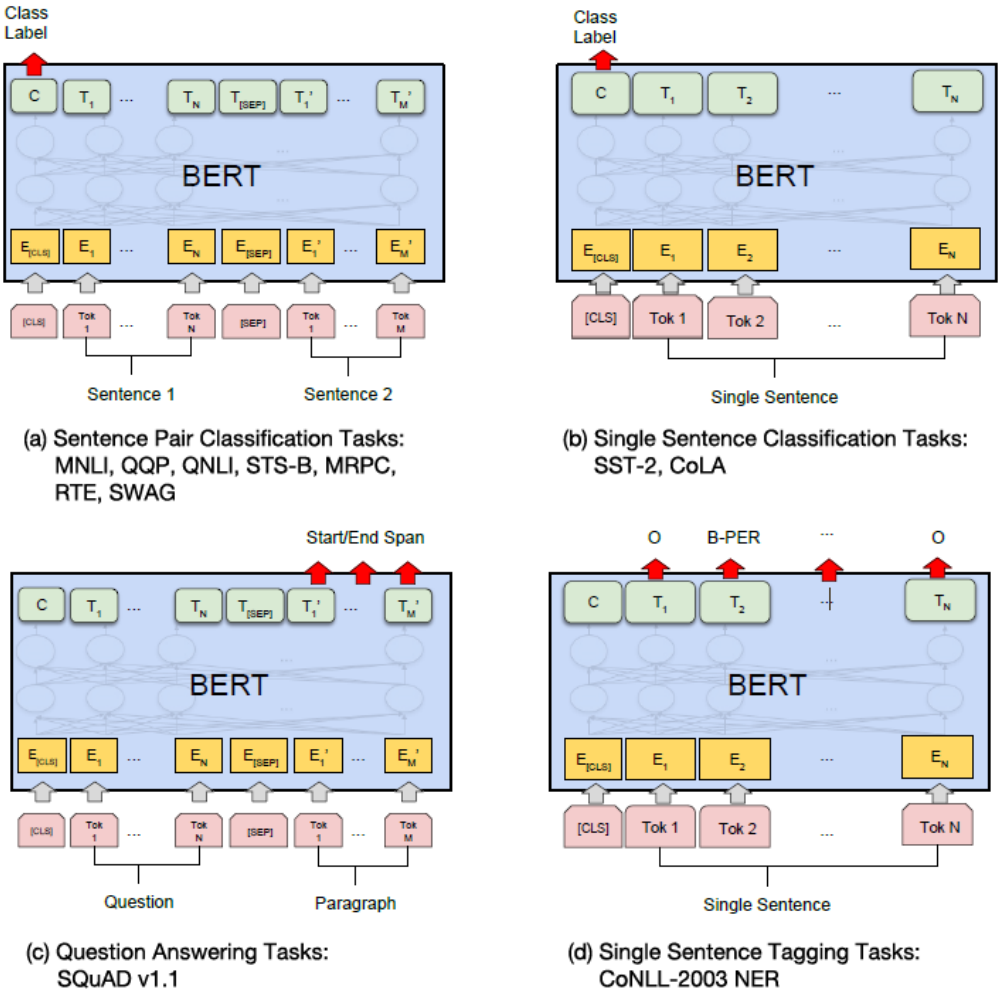

图 17 BERT 模型针对不同任务的微调[7]

在本课题中，我们将根据文本分类任务对已经经过预训练的中文 BERT 模型进行微调，并进行了如下的实验。

3.4 实验：单标签多类分类

实验采用的是 BERT-Base 的中文版本。相较于普通的 BERT-Base 模型，中文版本增加了 BasicTokenizer 和 tokenization.py 两个文件使得模型可以读取中文字符，并通过在中文字符两侧增加空格使得文本可以通过 WordPiece 算法进行 embedding。我们只需要在上述预训练模型的基础上进行微调，使得模型可以满足文本多分类任务的需要即可。

3.4.1 微调阶段使用的参数

--task_name：选择要使用的处理模式（Processor），包括 XNLI 模式，MNLI 模式，MRPC 模式，Cola 模式和本实验使用的自己编写的 Multilabel（多标签）模式。

--do_train：选择是否是要对模型进行微调。

--do_eval：选择是否对微调结果定时进行评估。

--do_predict：选择是否在微调结束后进行测试。

--do_export：选择是否导出 pb 模型。

--data_dir：选择数据集存储的路径。

--vocab_file：选择进行 tokenization 的词汇文件。

--bert_config_file：选择配置文件路径。

--init_checkpoint：加载上次的 checkpoint。

--train_batch_size：设置 batch_size 的大小，本课题受限于训练硬件条件，一般选择 4 或 8。

--learning_rate：设置学习率，一般选择 2e-5 或 3e-5，本实验中采用 2e-5。之所以选择 2e-5 这样比较低的学习率，是为了减轻灾难性遗忘（catastrophic forgetting）[39]现象带来的危害。灾难性遗忘是指在一个顺序无标注的、同种任务可能长时间不复现的任务序列中，AI 对当前任务 B 进行学习时，对先前任务 A 的知识会突然地丢失的现象。

--num_train_epochs：设置 epoch 数目，一般设置为 3.0。

--max_seq_length：最大能处理的序列/文本长度，最高设置为 512。这是由输入数据处理时需要进行 positional embedding 导致的。

--output_dir：选择模型输出路径。

--export_dir：选择模型导出路径。

3.4.2 数据预处理

本实验使用的是 THUCNEWS 中文新闻长文本标注数据集[40]和今日头条中文新闻短文本分类数据集[41]这两个数据集。

（1）短文本

今日头条中文新闻短文本分类数据集也是每行为一条数据，以_!_分割的个字段，从前往后分别是：新闻 ID，分类 ID，分类名称，新闻字符串（仅含标题），新闻关键词。在本实验中，我们只需要分类 ID 和新闻字符串，但是其他数据在预处理时也予以保留。

首先，我们把 txt 格式的文件转换成 csv 格式。采用按行读取的方式。由于 python 中 csv 模块在生成 csv 文件时自动用“,”分隔，为了避免引起错误，我们把文本中所有的“,”都替换成“，”（中文逗号）。然后再一行行写入 csv 文件。

为了让分隔符更独特，防止样例中出现混淆，转化为 csv 文件后，我们又将“,”分隔符转换成原来的“_!_”。在后面进行数据处理时，我们只需按_!_分隔，按照不同部分的 index 获取数据即可。

（2）长文本

THUCNEWS 中文新闻长文本标注数据集中每行为一条数据，以_!_分割的字段，从前往后分别是 类别 ID，类别名称，文本 ID，文本内容。长文本与短文本在预处理阶段的操作基本相同。唯一的区别在于，由于 BERT 模型有长度限制，样例最长为 512 个字，其中还包括一些特殊 token，在文本分类中，要包含开头的[CLS]和结尾的[SEP]，因此实际只能最多装 510 个字。而数据集中有些样例是长于 510 个字的。为了解决这一问题，我们提出了以下几种思路：

（1） 取文本的前 510 个字（前截断）

（2） 取文本的后 510 个字（后截断）

（3） 取文本的前 255 和后 255 个字拼接（前后截断）

（4） 采用文本提取方法提取出长度适合的文本（文本摘要）

（5） 把超过 510 个字的文本拆分成多个长度合适的文本（拆分法）

如果一个长文本的重要信息是在开头，可能方法（1）效果是比方法（2）要好。同理，方法（2）对信息点在结尾的长文本效果较好。使用方法（3），是综合方法（1）和方法（2），一般而言是好于单一的截断方式。但是这些方法，都会丢失一部分序列信息。在文本不是特别长的场景这三种方法效果可能不错。但如果是更长的文本，文本长度几千字，如果直接使用以上三种截断法，必然会丢失大量信息，造成准确率下降。

在某一些场景，可以尝试对原始文本进行过滤，降低有效文本的大小，即方法（4）。关键点是如何筛选出有效句子和有效信息。这就涉及到自然语言理解任务中的文本摘要任务。

自然语言处理（NLP）中有两种文本摘要生成方法：抽取式和抽象式（生成式）。抽取式摘要的方法是，抽取一段文本中表示重点内容的单词子集，并结合起来生成摘要。抽取式摘要通常需要衡量基本句子成分的权重，并根据权重结果生成摘要。不同类型的算法和方法均可用于衡量句子的权重，之后根据各成分之间的关联性和相似性进行排序-并进一步将这些成分连接起来以生成摘要。抽象式摘要的方法是，使用高级深度学习方法（advanced deep learning technique）用于解释和缩写原始文档，生成表示源文本中最重要信息的新短语和句子，所以这些抽象式算法有助于克服抽取式摘要中的语法不准确问题。但是在本实验的情景下，由于 BERT 模型会学习单词与前后单词，句子与句子之间的关系，从定性的角度讲，抽象式摘要方法中生成的新短语和句子可能会为微调过程带来噪音，会比抽取式的摘要方法带来更大的误差，而且 BERT 模型的训练方法会进一步放大这种误差，因此本实验采用抽取式摘要方法进行过长文本的摘要。

最终本实验采用的是 TextRank 算法[42]。TextRank 算法是一种抽取式的文本摘要方法，基本原理是基于图的排序，把文本分割成多个组成单元（在本实验中每个单元就是一个句子），构建节点连接图，边的权重就是句子之间的相似度，对生成的图进行循环，计算句子的 TextRank 值，最后抽取一定数量的，TextRank 值高的句子进行组合，生成文本摘要。具体流程为： 第一步是把所有文章整合成文本数据，接下来把文本分割成单个句子，然后，我们将为每个句子找到向量表示（词向量）。计算句子向量间的相似性并存放在矩阵中，然后将相似矩阵转换为以句子为节点、相似性得分为边的图结构，用于句子重要性计算。最后，一定数量的排名最高的句子构成最后的摘要。在本实验的数据预处理过程中，我们先运行 Textrank 程序把数据集中的长文本进行摘要，然后再进行微调。

方法（5）是将一个整段的文本拆分为多个部分，每一个部分的长度小于 510。拆分可以暴力地通过 510 的大小进行划分，或者通过断句的方式，将相邻的句子放入一个部分。这种方法将所有序列都放入模型之中。考虑到了全局的信息，对文本很长且对截断敏感的任务有较好的效果。但有一些缺点：比如性能较差，原来截断法需要解码一次，方法（5）需要解码多次，篇章越长，速度越慢。而且各个部分之间的联系会丢失，可能会出现 bad case。在本实验情境下，由于 BERT 模型会学习句子之间的关系，那么暴力的通过 510 字的大小进行划分就会为模型训练带来不便。因此本实验采用断句的方式，把过长的样例文本分成同类型的几个长度合适的样例。

上述分析都是定性的分析，真正的实际效果将在后续的实验中进行对照。本实验中，我们使用以上五种方法分别处理相同的长文本数据集，然后分别进行微调评估结果。

3.4.3 训练流程

由于我们已经下载了经过预训练的 BERT 模型，所以只需要引用三个模块：modeling，

optimization 和 tokenization。Modeling 模块提供创建模型的接口，optimization 模块优化训练过程，tokenization 模块对输入文本进行单词化处理。

在微调过程开始之前，首先读取输入的所有参数，根据参数调用函数匹配相应的 checkpoint 并配置 BERT 模型。同时还要检查最大序列长度是否超过了 512，并检查 do_train, do_predict, do_eval 三个参数至少有一个为 TRUE。最后创建输出文件夹，根据 task_name 参数选择处理方式（本实验中选择 multilabel processor）并读取所有分类标签的列表。

（1）训练阶段

如果 do_train 参数为 TRUE，则说明我们要进行微调训练。首先要把文本转化为 BERT 模型的输入。我们首先定义了 BERT 模型输入的格式。分为四部分：guid 代表输入样例的 id, text_a 代表输入样例的标题, text_b 代表输入样例的正文, label 代表输入样例的标签。这四部分共同组成 object 类型的样例。

为了把存储在 csv 文件中的数据集转化为上述样例的格式，我们首先从输入参数 data_dir 的目录下，读取所使用数据集的 csv 文件，然后调用创建输入样例函数，循环读取每一行数据，并根据训练集，测试集和开发集三个不同的数据集设置 id，格式为“数据集类型”-“行数”，这样每条数据就拥有了不会相同的 id。由于数据集里的数据中，标题和正文是使用空格隔开的，无法进行分离（因为标题和正文中同样有可能有空格），所以把标题和正文全部放入 text_a，text_b 为空，然后获得标签。

然后我们需要把输入样例的集合转化为 TFRecord 格式的文件（文件名为 train.tf_record）。首先，把 text_a 和 text_b 转化为 tokens_a 和 tokens_b（即上文提到的词嵌入 token_embedding 过程）。把两者长度相加与最大序列长度比较，检查输入的有效性。然后给 tokens_a 和 tokens_b 的第一个单词前添加[CLS]，表示是文本的第一个词，句子和句子之间添加[SEP]，区别一条文本中的两个句子（在本实验中即区别文本的标题和正文，把标题的 segment_ids 设为 0，把正文的 segment_ids 设为 1）。

然后设置训练数据的掩码，在本实验中，文本中的单词的掩码设置为 1，而刚才添加的单词（如[CLS]）的掩码设置为 0，但是由于我们只将文本中的单词进行掩码操作，所以掩码序列中的元素全为 1。

最后把得到的单词序列，掩码序列用 0 补齐至最大序列长度。然后把 input_mask，segment_ids 等训练特征封装成 features 对象返回。在最终输入模型前，还需要经过切片，这样就把数据集里的文本转化成 BERT 模型可以读取的样例。

调整完输入数据后，我们根据样例的长度，训练回合数目和批尺寸大小计算训练的步长。计算方式为步长=样例长度/批尺寸*训练回合数目。

这些工作完成以后，我们就开始加载 BERT 模型。由于 BERT 模型是用 Estimater 封装的。所以我们首先创建 model_fn 对象和 estimater 对象，调用 train 函数。这样我们就可以利用样例中的 features 对象的参数创建模型并开始训练。我们使用 create_model 函数创建初始模型，然后加载上次训练 checkpoint。最后调用 EstimatorSpec 函数输出 loss。

（2）评估阶段

EVAL 阶段就是对训练模型的表现进行评估，包括准确率，loss 等指标。EVAL 阶段与 TRAIN 阶段在数据集处理，方法调用上没有区别，只是用 evaluate 函数代替了 train 函数。因此在加载模型时 estimater 的模式为 EVAL，此时就不再是用 EstimatorSpec 函数输出 loss，而是调用 metrics.accuracy 和 metrics.mean 函数输出准确率和 total_loss。

（3）预测阶段

PREDICT 阶段和以上连两个阶段都不同，因为数据集是测试集，所以不像训练和评估阶段时文本和所属的分类都已知，而是模型要根据文本判断所属的类型，然后把所属的类别进行输出，与测试集中的标准数据相比较，得出准确率。因此，在数据集处理上，我们

把所有样例的标签全部设置为 0，然后进行处理。加载模型时，使用 predict 函数，用 EstimatorSpec 函数输出模型做出的预测和该预测的概率，输出到名为 test_results.csv 的文件中。

在微调过程结束后，我们使用 output.py 文件手动比较 test_results.csv 和测试集中的数据，计算准确率。

（4）导出阶段

EXPORT 阶段的作用，是把模型导出为 pb 格式的文件，方便接下来开发应用时模型的持久化。原始的 BERT 模型主要设计为单次运行的目的，如果把 do_predict 参数设置成 True，也可以进行预测，但输入样本是基于文件的，并且不支持将模型持久化在内存里进行多次调用，会造成不必要的麻烦，影响应用的性能。因此需要自己改写代码，达到两个目的：

（1）允许将模型加载到内存里。允许一次加载，多次预测。

（2）允许读取非文件中的样本进行预测。譬如从标准输入流读取样本输入。

BERT 模型使用 Estimater 封装，但是遗憾的是 Estimator 对象并不支持一次加载，多次预测。因此需要使用 export_saved_model() 方法把 estimator 重新导出成 tf.saved_model。执行后，会有一个时间戳命名的模型目录。这样之后我们就不需要 estimator 对象了，可以直接调用 predictor.from_saved_model 函数自行从刚刚的模型目录加载模型，在应用模型时会方便许多。

3.5 实验：多标签多类分类

在多标签分类问题中，训练集由实例组成，每个实例可以被分配到表示为一组目标标签的多个类别，并且任务是预测测试数据的标签集。例如：文本可能同时涉及任何宗教，政治，金融或教育，也可能不属于任何一种。电影可以根据其摘要内容分为动作，喜剧和浪漫类型。有可能电影属于 romcoms （浪漫与喜剧）等多种类型。

在多类分类中，每个样本被分配给一个且仅一个标签（即单标签多类分类）：水果可以是苹果或梨，但不能同时是两者。在多标签情况下，每个样本可以属于一个或多个类。

解决多标签分类问题，由两种思路：

第一，把多分类问题拆分成多个二分类问题。这种方法在标签量小，模型不复杂的时候可以使用。但是在实际应用中，标签数量一般会超过 5 个，这意味着一条数据我们需要进行至少五次输入模型的过程，而每次输入都需要加载模型，这带来了巨大的时间和空间开销，不适合应用在舆情平台这样的应用中。

第二，把单标签分类模型进行改造，使其能够满足多标签分类的需要。这种方法可以有效解决上述的问题，只需一次输入，一次加载模型，就可以直接得到标签数据。但是与单标签分类相比，多标签分类进行预测时会有多个标签概率较高。而原模型的输出层采用的 softmax 函数的输出值相互关联，不能满足多标签分类的需要。

本实验在准备阶段存在一个问题，由于没有经过很好地标注的中文多标签文本分类数据集，所以本实验只能采用英文数据集，同样的，BERT 模型也不再使用 BERT-Base 的中文版本，使用 BERT-Base 预训练模型，最终本实验数据集采用恶毒评论分类挑战[43]，除了序号和原始文本以外，每行数据都包含了 6 个维度的标注，分别是：toxic（恶毒），severe_toxic（非常恶毒），obscene（污言秽语），threat（威胁），insult（侮辱），identity_hate（憎恨）。如符合某一项标签，则标注为 1，否则标注为 0。

多标签分类与单标签分类在代码逻辑上并没有什么不同，主要区别有两个：

（1）样例数据输入时，单标签只有一个 label，故输入对象中的标签属性为一个字符串，而多标签多类分类中样例数据有多个标签，如：[0,0,0,0,0,0]。因此输入对象中的标签属性是作为一个数组输入的。

（2）在文本分类中将分类器的原始输出值映射为概率的方式有 softmax()，sigmoid()两种。在简单的二进制分类中，两者之间没有太大的区别，但是在多标签分类的情况下，sigmoid 允许处理非独占标签（也称为多标签），而 softmax 处理独占类。Sigmoid 函数会分别处理各个原始输出值，因此其结果相互独立，各标签概率总和不一定为 1。相反，softmax 函数的输出值相互关联，其概率的总和始终为 1，增大某一类别的概率，其他类别的概率必须相应减少。在多标签多类分类实验中，显然 sigmoid 函数是更适合本实验的映射方式。因此把原模型中 softmax 函数，替换为 sigmoid 函数，同时在计算 loss 时，为计算每个示例损失，我们使用 sigmoid_cross_entropy_with_logit 函数测量离散分类任务中的概率误差，其中每个类是独立的而不是互斥的。

进行以上更改后，我们就可以开始训练。

3.6 敏感词检测技术研究与实现

随着信息时代的发展，涉及政治，色情，暴力等因素的敏感词汇也层出不穷。在许多对外公共场合下，有些内容是要经过审查才能显示的。理论上讲，只要涉及用户输入的地方，都需要进行文本校验，以保证敏感词汇不被展示在网络上。所以我们需要一个有效高效的敏感词检测手段，在涉及文本输入的时候进行快速的检测。既不影响用户的使用体验，又能准确地过滤非法信息。在网络审查初期，都是通过人工审核，这种审核方式虽然准确，但与网络上文本产生的速度相比，其效率就显得过于低了。因此，自动化的敏感词检测方法的需求越来越强烈。

3.6.1 DFA 算法

DFA(Deterministic Finite Automaton)，即确定有限状态机，基本工作原理是通过事件和当前的状态得到下一个状态。在敏感词检测的任务中，我们可以把字符看做状态，把字符间的前后联系视作事件，这样就把字符的比较转化为状态的转移。具体的算法原理为：

首先以敏感词库中的词汇建立一个确定性的树形有限状态机。比如，假设敏感词库里有 abc，abd，ef 三个敏感词，我们就可以建立 a->b->c ，a->b->d ,e->f 的树形结构。然后以待检测文本作为该有限状态机的输入，使状态机进行状态的转换，当到达某些特定的状态时，说明发生文本匹配，即在待检测文本中检测到了敏感词汇。假设待检测文本为 abcdefg，首先检查 DFA 中是否有 a 这样一个起始状态，若存在，则检查第二个字符 b 是否是 a 的下一个状态，若是，再检查下一个字符 c 是否是 b 的下一个状态，结果为是。由于 c 已经是此确定有限状态机这条分支的最后一个状态，所以我们可以判定待检测文本中存在敏感词 abc。这样就完成了一次检测的过程。

3.6.2 算法实现

敏感词检测功能的代码分为三个部分：

add()和 parse()函数，负责建立状态机，以多维数组的形式储存敏感词库中的敏感词。首先，把关键字去除首尾空格和换行，然后遍历关键字的每个字，如果这个字已经存在字符链的 key 中就进入其子字典，如果不在就建立新的子字典。

Filter()函数，负责根据待检测文本，以循环的方式查询已建立好的多维数组，得出结论是否匹配。假如匹配，则用长度与敏感词长度相同的连续占位符代替原有的敏感词，并把处理好的文本返回。

主函数 main()，负责处理输入输出，调用 parse()和 filter()，并输出整个敏感词检测的流程所用的时间。

3.7 本章小结

本章简单分析了现有文本分类方式中存在的问题，并基于 BERT 模型进行优化与改进，最终进行实验。本章详细介绍了 BERT 模型的预训练方法及微调过程，详细解释了微调过程各阶段的具体流程，最后在长文本单标签多类分类实验中探究了截断、拆分、文本摘要等方法，设计了多种长文本预处理方法，并分别进行了实验对这些处理方法进行了比对。

第四章 系统设计与开发

随着移动互联网的飞速普及,网络给民众带来了开放、便捷的信息共享和发布平台,人们在网络上发表意见、分享情绪和展示态度,其中既有对社会事件的发展有积极作用的信息,也有负面、消极的信息。网络平台的开放性和隐蔽性使得网络平台上的舆论一点点影响民众的意识形态。因此,对舆情信息的及时准确的监控，分析和汇报，对维护社会稳定、促进国家发展具有重要的现实意义。本课题紧紧围绕舆情平台，把上文实现的文本分类和敏感词检测技术应用到平台中去，完善舆情系统的功能，发挥算法准确高效的优势，使训练好的模型和算法在实际应用中发挥作用。

4.1 舆情系统介绍

舆情平台的主要功能包括：实时获取信息平台的舆情信息、快速检索舆情信息文本、快速发现舆情热点并进行分析与预警，生成舆情表使用户更直观全面了解舆情信息等。舆情系统的工作流程可以划分为五个阶段：获取数据、处理数据、存储数据、逻辑业务和用户交互。

数据获取层是整个舆情系统中最为基础的模块，其负责舆情系统的原始数据采集，通过各种网络爬虫对不同平台（如微博、微信、新闻网页、论坛等等）内容进行爬取，对多源数据进行归一化处理，本项目使用基于中文密度的正文抽取算法和 Scrapy 爬虫框架[44]，作为生产者将原始数据以 JSON 格式存入消息中间件 Kafka 中，供数据预处理层消费。

数据处理层紧接着数据获取层,是 Kafka 的消息消费者。其作用是将爬虫爬取到的 JSON 格式数据进行进一步的处理，如情感分析、敏感信息抽取、文本摘要生成等。爬虫作为消息中间件的生产者（Producer），后续的数据处理程序作为消息中间件的消费者（Consumer），同一类消费者属于同一个消费者组（Consumer Group）。在 Kafka 中，我们为不同的数据源创建相应的话题（微博、微信、新闻等等），在物理上不同话题的数据分开存储，不会相互干扰，并且便于区分。

数据存储层支撑起整个舆情系统中的数据流通。由于现在已经是一个海量信息的时代，舆情系统的数据量必然是巨大的，所以系统采用 HBase、Solr、MySQL、Redis 多样的结构来进行信息存储，不同的数据存储工具适合不同的场景，结合使用提升系统的吞吐率。其中 HBase 负责存放爬虫爬取的网页数据，Solr 负责构建舆情数据的全文索引，Mysql 负责舆情系统中结构化的业务数据存储，Redis 负责分布式多线程爬虫爬取 URL 队列的去重。

业务逻辑层是舆情系统功能的整合，包括全文检索、统计分析、数据维护等等。业务逻辑层负责将数据存储层的数据处理成为用户需要的特定格式，交付给下一层交互层。

交互层是直接与用户交互的层级，其作用主要是提供用户需要的界面，将业务逻辑层提供的数据可视化的展示给用户，提升用户体验。全部系统架构如图 18 所示。

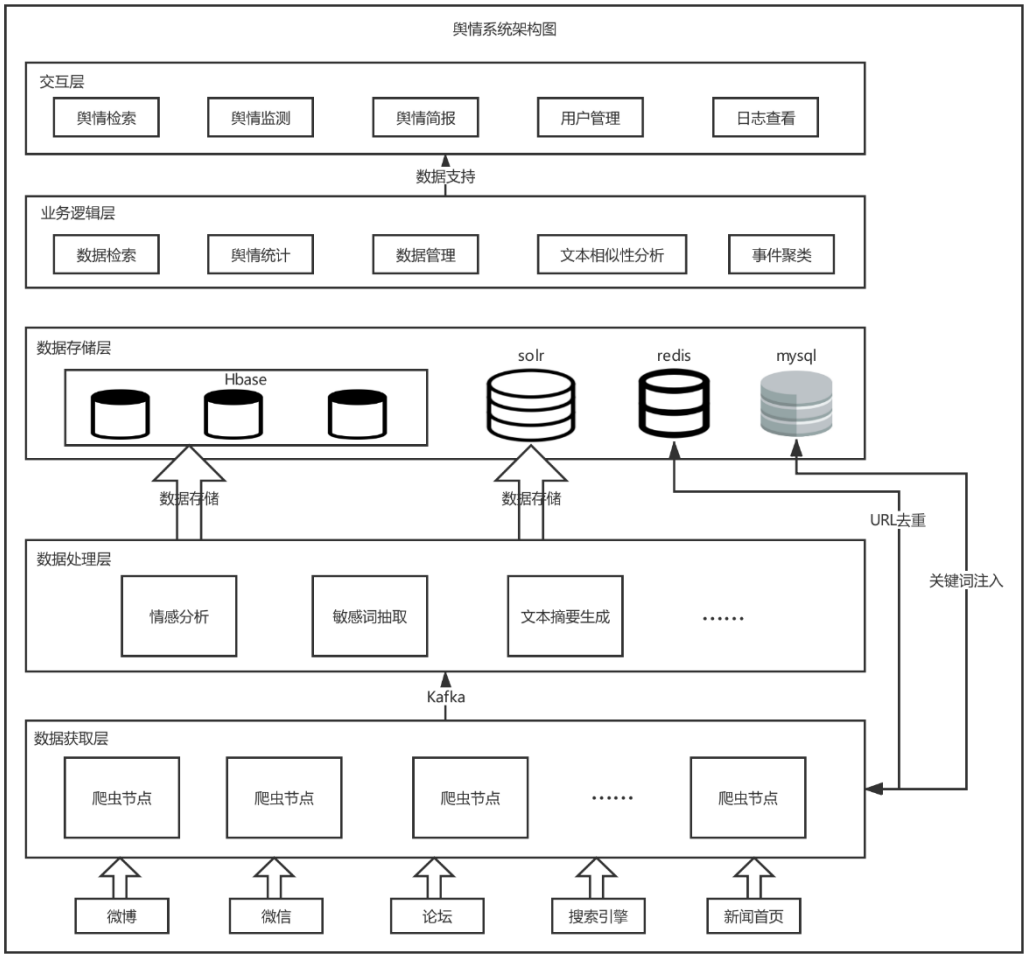

图 18 舆情系统架构图

在现有的舆情平台的基础上，我们将把上文训练的文本分类模型和敏感词检测算法应用到舆情平台中，使得我们的研究成果可以在实际应用中发挥作用。

4.2 功能模块设计

为了使文本多分类和敏感词检测功能能够更好的集成到平台中，实现这两个功能的应用价值，我们将这两个功能进行封装并结合舆情系统来使用这两个功能模块并使这两个功能可以更方便地迁移到别的平台。

4.2.1 文本分类模块

作为本文研究的核心，文本分类是舆情系统有效高效处理舆情文本的关键步骤。在第三章，我们已经训练得到了文本分类的预测模型，但是仍有问题需要解决。在原始 BERT 模型中，我们只能把输入数据以文件的形式进行分类预测，并且每次运行都需要重新加载一次模型，这样的工作模式会导致效率低下，无法满足舆情系统数量大，实时监控的功能需求。

为了解决上述问题，我们不再把模型导出为 ckpt 格式，而是导出为 pb 格式模型并使用

predictor.from_saved_model 函数直接加载模型并持久化，使用在 predict 阶段定义的的 predict_fn 变量，就可以直接进行预测。在此系统中，我们从标准输入流读取问题样本，并预测分类。即先按照("id", question, None, 0)的格式定义变量 predict_example，再获取样例的 feature。最后进行预测，获得预测结果 prediction。变量 prediction 中储存着每个标签对应的概率，我们先获取这些概率，然后寻找概率最大的一项，在标签列表中按相同的 index 获得标签。这样就完成了一次完整的预测过程。

为了降低运行成本，我们改写原来的文件，把加载模型和进行预测的步骤尽可能地从原模型的复杂代码中剥离出来。首先我们可以减少参数的个数，因为预测过程不涉及训练部分，所以训练相关的参数可以全部去除，包括：数据集存储目录，批尺寸，学习率，训练回合数等。然后，我们也可以去除原代码中创建的 model_fn 和 Estimator 对象，因为我们有新的加载模型的方式。最后，我们去除不必要的参数检查，比如 do_train，do_eval，do_predict 和 do_export，这样可以去掉不必要的数据处理步骤。

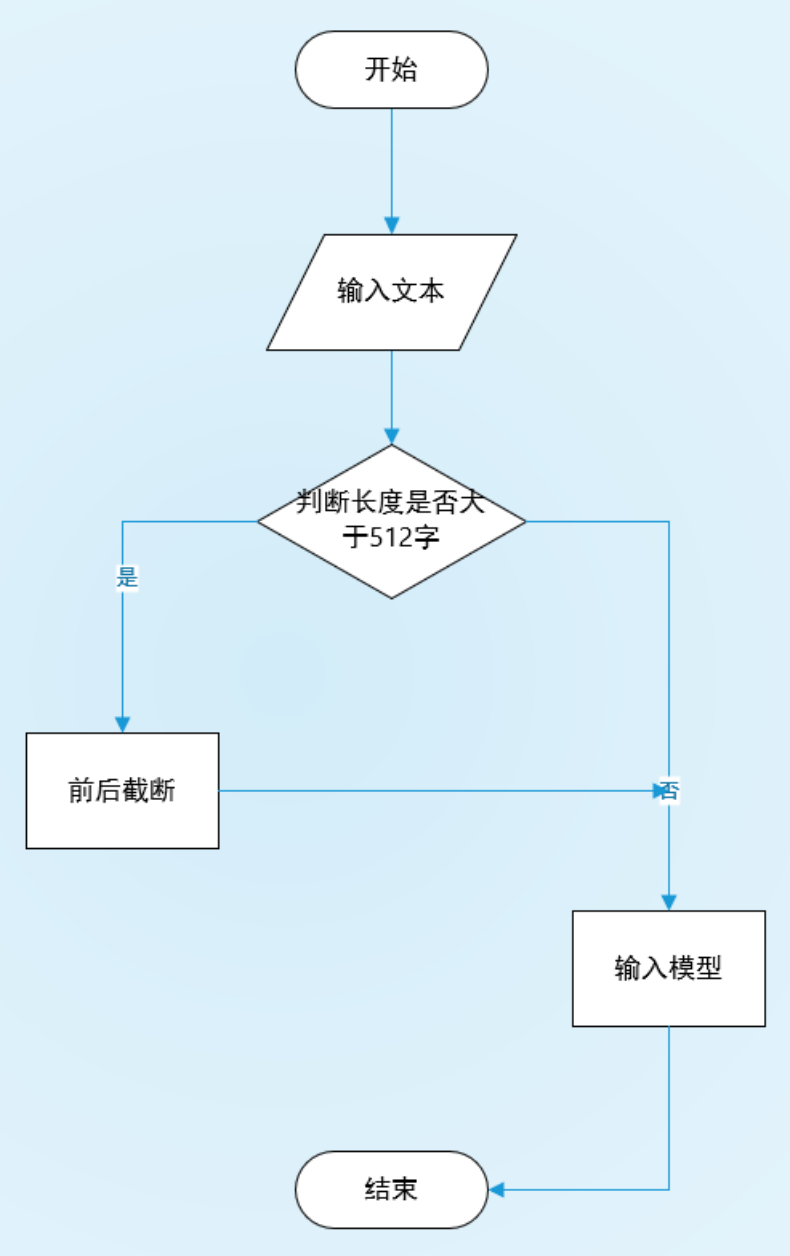

图 19 文本分类模块流程图

本模块基于 flask 框架实现了一个接口，先加载模型再进行文本分类，大大降低了模型载入的时间。由于需要进行文本分类的文本既有短文本（如微博），也有长文本（如新闻），所以本模块在处理输入数据时做了不同的处理，短文本直接作为输入，长文本进行前后截取操作再输入模型。输出是模型预测的类别 ID。该模块的工作流程如图 19 所示。

4.2.2 敏感词检测模块

及时准确地发现带有敏感词的舆情文本对于舆情系统非常重要，所以敏感词检测模块也是数据处理时的重要模块之一。在第三章我们已经实现了 DFA 算法，但是在本实验中，我们不止希望能够鉴别并过滤敏感词，而且我们建立了不同种类的敏感词库，希望能够得到敏感词具体的分类。这就需要在原本的代码中加入一个分类的步骤，解决这一问题有两种思路：

第一，可以给每一个敏感词加上一个标签，标注该敏感词的敏感原因，假如匹配到了该敏感词，则输出该词的标签。这样做比较直接，但是会造成内存空间的占用变大，假如敏感词数目比较大，这种方式会影响应用的性能。

第二，把同种类的敏感词分类，分次多次建立状态机。这样的方法可以有效节省内存空间，但代价是处理时间的增加。这种方法在敏感词种类比较多的时候不占优势。但是对于敏感词种类不多但每个类别中的数目较多的敏感词库比较有优势。

由于本次实验采用的敏感词库有五种分类，每个分类下有几百个长度不同的词汇，最终我们选择方法 2 对敏感词库进行处理。但是要说明的是，两种方法在这样的数据量下，处理时间上的区别不大，在空间上的优势是我们选择方法 2 的主要原因。

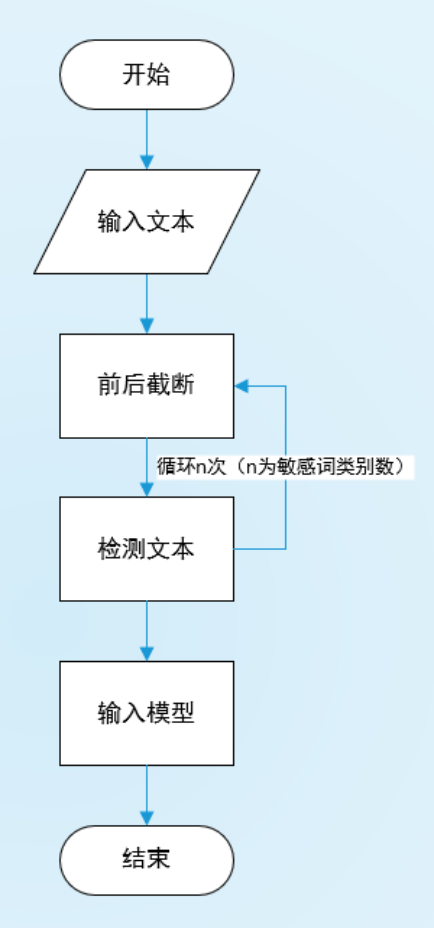

图 20 敏感词检测模块流程图

敏感词检测模块的工作流程如如图 20 所示。首先将敏感词词表构建成一颗字典树载入内存，然后使用 DFA 算法对文本进行匹配。由于敏感词表的字典集合已知，我们在每次处理时可以直接载入内存，无须进行重复的构建操作，降低了算法的时间复杂度，满足了对文本进行快速检测的功能需求。

4.3 系统功能验证

本系统为了展示文本多分类和敏感词检测的功能，进行三个功能的展示：上传文章，按标签搜索文章，新增分类标签。

4.3.1 上传文章并归档

上传文章归档功能模拟的是用户编写一篇文章上传至平台的操作。这一功能主要展示敏感词检测和文本多分类的实际应用。首先用户在输入框中输入文本，编辑标题以后，点击提交。此时后台先进行敏感词检测操作：获得用户的文本信息，调用 DFA 函数。假如用户的输入信息中存在敏感词，则会重定向到新的页面，页面中展示敏感片段并展示敏感原因。假如用户的输入信息中不存在敏感词，则会调用文本多分类函数，加载模型，对用户的输入文本进行预测，得到预测结果后，把文本按照“类别”—“标题”—“正文”的格式存入数据库（如图 21）。在系统的首页可以进行展示。

图 21 上传文本界面

图 22 分类结果展示

4.3.2 按标签搜索文章/标题

按标签搜索文章/标题功能展示的是文本多分类的应用，模拟的是假如管理员只有一些没有标签的文本（假设这些文章不存在敏感词），但是需要获得对应标签下的所有文章的操作。我们把这个功能分为两步：先分类，再搜索。我们假设待分类文本存储在另一个文件中。我们按行读取文本，执行文本分类函数，然后把获得对应标签的文本按照“类别”—

“标题”—“正文”的格式存入数据库。在搜索时，我们从前端获取要搜索的标签，然后在数据库中按“类别”条目进行搜索，获得对应的结果展示在前端中。而管理员也可以选择搜索文章内容或是搜索标题，相对应的就是长文本和短文本的分类。

图 23 批量上传界面

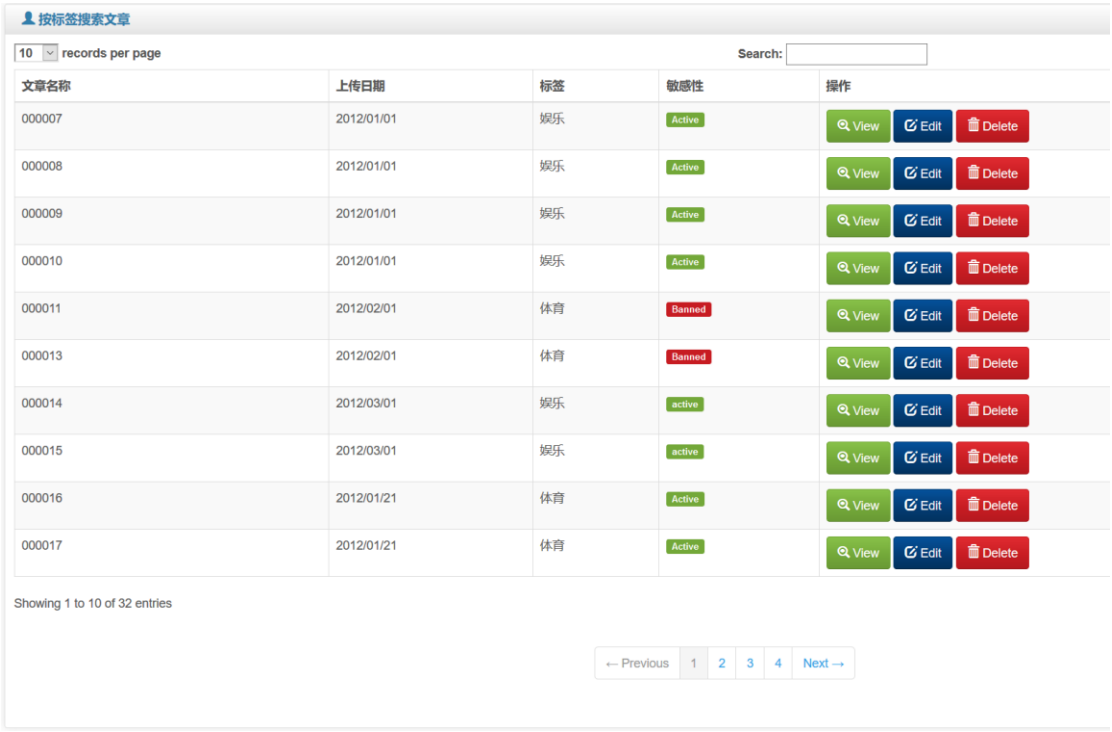

图 24 按标签搜索文章

4.3.3 新增标签分类

新增标签分类功能是为了应对已有的类别不能满足实际应用的需要的情况，我们需要根据实际情况新增标签。但是由于分类器是一个深度学习模型，不经过训练（即微调）是无法处理新标签的文本的。因此，我们需要在原来的数据集中加入新标签下的文本，对 BERT 模型进行重新微调。当然由于训练时间长，训练条件要求苛刻，所以我们不展示具体的训练过程，只针对微调阶段的数据预处理过程进行展示。为了保证数据集的质量，获得高准确率的分类模型，我们不能单纯的把新文本插入原数据集中，需要打乱训练集数据的顺序。由于本实验的训练集数据量为短文本 26 万条，长文本 33436 条，故选择长文本训练集进行添加新标签的操作，简化系统的流程。在旧模型中无法分类的文本，在训练后的新模型中

能够得到准确的分类，就实现了新增分类标签的效果。

4.4 本章小结

本章首先分析了舆情系统的具体需求，简单阐述了舆情系统的工作流程和整体架构，然后对本课题实现的功能模块的设计进行具体解释，并通过前端页面对文本分类和敏感词检测功能进行演示，实现了舆情系统的一条完整的业务流程，证明了上文中训练的文本分类模型和敏感词检测模块的有效性。

第五章 实验结果

5.1 实验环境

本课题所进行的实验基于 Linux 环境（CentOS7.7.1908），中央处理器 CPU 型号为 Intel 至强 E5-2650，搭配 128G 内存和 16G 显存的 NVIDIA P100 显卡。本实验使用的 CUDA 为 10.0.130 版本，CUDNN 为 7.6.5 版本，Tensorflow 为 1.14 版本。

5.2 短文本单标签多类分类实验

5.2.1 数据集数据分析

本课题实验采用的短文本数据集的训练集共包含 267881 条数据，开发集 57404 条数据，测试集 57403 条数据。该数据集的样本共有 15 种分类标签，每个样本只能属于一种标签，每种标签在训练集，开发集和测试集中的分布如表 1 所示：

表1 短文本多类分类数据集数据分析

类别	类别 ID	训练集占比	开发集占比	测试集占比
故事	100	0.0164	0.0159	0.0164
文化	101	0.0737	0.0706	0.0736
娱乐	102	0.1029	0.1036	0.1020
体育	103	0.0981	0.0981	0.0981
财经	104	0.0704	0.0713	0.0715
房产	106	0.0461	0.0460	0.0462
汽车	107	0.0935	0.0949	0.0918
教育	108	0.0704	0.0700	0.0726
技术	109	0.1084	0.1088	0.1086
军事	110	0.0650	0.0650	0.0668
旅行	112	0.0561	0.0544	0.0567
世界	113	0.0703	0.0723	0.0680
股票	114	0.0008	0.0009	0.0007
农业	115	0.0503	0.0506	0.0511
游戏	116	0.0768	0.0768	0.0750

数据样例：

6552431613437805063_!_102_!_news_entertainment_!_谢娜为李浩菲澄清网络谣言，之后她的两个行为给自己加分_!_佟丽娅,网络谣言,快乐大本营,李浩菲,谢娜,观众们

5.2.2 不同实验参数对比

在实验中设置的诸多参数中，我们主要对批尺寸（batch size）、学习率（learning rate）和回合数（number of epochs）三个参数进行分析优化。对于批尺寸，如果参数设置比较小（如 2，4，8），每次输入的样例数量少，会增加训练时间，也会给模型收敛带来困难，但是对训练环境的硬件（尤其是内存）要求比较低。如果参数设置比较大（16，32，64 等），会占用更大的内存空间，但是训练模型更容易收敛，训练时间也更短。由于批尺寸为 32 时 16G 显存不能满足训练需要，所以本实验研究了批尺寸为 8 和 16 两种情况。在本实验的实验结果中，批尺寸为 8 或 16 对于模型的表现影响不大。

在学习率的选择上，主流的选择为 2e-5，3e-5，5e-5 三种，学习率过大会导致损失函数不断震荡，模型不易收敛，学习率设置得过小会使得模型收敛速度过慢。本文选择 2e-5，原因已经在第三章进行了阐述。我们经过实验，发现 2e-5 的学习率虽然小，但足够让模型得到很好的收敛。

最后，过大的训练回合数会导致模型过拟合，过小的训练回合数则无法完全学习训练集文本的完整信息。BERT[7]模型的论文作者在文章中给出了比较适合的回合数，为 3 和 4。经过实验，参数设置为 3 和 4 的区别不是很大，但在有些情况下，训练回合数为 4 的模型表现比训练回合数为 3 的模型更好。考虑到提升训练回合数带来的训练时间的成本，我们有理由得出结论：训练回合数选择 3 或 4 对于模型表现影响不大。

表2 短文本多类分类测试结果

学习率	批尺寸	训练回合数	准确率
2e-5	8	3	0.8922
2e-5	8	4	0.8954
2e-5	16	3	0.8978
2e-5	16	4	0.8966

5.2.3 不同模型结果对比

经过实验，短文本单标签分类准确率最高为 89.78%，达到了目前 BERT-Base 模型的基准值。

表3 短文本多类分类测试结果对比

模型	准确率(%)
BERT-Base	89.78
ERNIE-base	89.83
XLNet-mid	86.26
ALBERT-xlarge	88.30
RoBERTa-wwm-ext	89.79

相比于 ERNIE-base[45]、XLNet-mid、ALBERT-xlarge[46]、RoBERTa-wwm-ext 模型，本课题训练得到的短文本单标签文本分类模型达到了目前主流文本分类模型的平均水平，相较于表现更好的 ERNIE-base 模型和 RoBERTa-wwm-ext 模型，准确率相差不超过 0.05% ，完全可以在实际的舆情系统中进行应用，达到了课题要求的标准。

5.3 长文本单标签多类分类实验

5.3.1 数据集数据分析

本课题实验采用的长文本数据集的训练集共包含 33436 条数据，开发集 4180 条数据，测试集 4180 条数据。该数据集的样本共有 14 种分类标签，每个样本只能属于一种标签，每种标签在训练集，开发集和测试集中的分布如表 3 所示：

表4 长文本多类分类数据集数据分析

类别	类别 ID	训练集占比	测试集占比	开发集占比
体育	0	0.1569	0.1645	0.1540
娱乐	1	0.1113	0.1069	0.1102
家居	2	0.0390	0.0349	0.0425
彩票	3	0.0086	0.0102	0.0110
房产	4	0.0243	0.0232	0.0217
教育	5	0.0489	0.0557	0.0540
时尚	6	0.0161	0.0133	0.0169
时政	7	0.0767	0.0703	0.0700
星座	8	0.0043	0.0031	0.0050
游戏	9	0.0292	0.0270	0.0303
社会	10	0.0610	0.0590	0.0607
科技	11	0.1937	0.1990	0.2000
股票	12	0.1849	0.1875	0.1799
财经	13	0.0444	0.0447	0.0430

数据样例：

11_!_科技_!_493337.txt_!_爱国者 A-Touch MK3533 高清播放器试用　　爱国者 MP5 简介:　　"爱国者"北京华旗资讯，作为国内知名数码产品制造商。1993 年创立于北京中关村，是一家致力于.....

5.3.2 不同处理方法对比

表5 长文本多类分类测试结果

处理方法	准确率（%）	Loss
前截断	95.90	0.2299
后截断	93.94	0.3222
前后截断	96.02	0.2258
文本摘要	93.95	0.3310
拆分法	91.38	0.4706
基准值	95.35	无

长文本单标签分类准确率最高为 96.02%，比基准值高了 0.67 个百分点。这样的提升来自于前后截断的方法，因为前后截断非常适用于新闻类的文本。从表中我们可以看出，使用文本前后截取的方法效果最好，是因为前后截断非常适用于新闻类总分式的文本，前截断和后截断的方法都不够好。虽然拆分的方式没有截断法准确率高，但是文本拆分所保留了最完整的文本信息，处理文本的损失最少。而文本摘要的方法，由于在摘要的长度（要保留的句子数目）上不容易确定一个固定的数值，所以得到的摘要后的文本长度受单句长度影响较大，造成的后果就是文本长度不稳定，造成预测准确率偏低。

综上所述，最适合舆情系统的长文本处理方法，是前后截断的方法。

5.3.3 不同模型结果对比

从表 6 中可以看出，相比于 ERNIE-base、RoBERTa-wwm-large、XLNet-mid、ALBERT-xlarge 等主流深度学习模型，本课题基于 BERT 的长文本单标签文本分类模型有了很大程度的提升，比其中最好的 RoBERTa-wwm-large 模型准确率提升了 0.1%。这样的数据可以证明，本实验设计并训练的 BERT 模型取得了超越当前主流模型的表现。

表6 长文本多类分类测试结果对比

模型	准确率(%)
BERT-Base	96.02
ERNIE-base	94.90
RoBERTa-wwm-large	95.93
XLNet-mid	94.54
ALBERT-xlarge	95.45

5.4 多标签多类分类实验

5.4.1 数据集数据分析

表7 多标签多类分类数据集数据分布

维度	训练集	测试集
toxic（恶毒）	0.248	0.239
severe toxic（非常恶毒）	0.147	0.167
obscene（污言秽语）	0.225	0.235
threat（威胁）	0.120	0.107
insult（侮辱）	0.197	0.127
identity hate（憎恨）	0.104	0.114

训练集包含 4000 条数据，测试集 1000 条数据。每条样例数据都包含了 6 个维度的标注，分别是：toxic（恶毒），severe toxic（非常恶毒），obscene（污言秽语），threat（威胁），insult（侮辱），identity hate（憎恨）。每个维度在训练集和测试集中的分布如表 7 所示。

数据样例：

0000997932d777bf,″Explanation Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27",0,0,0,0,0,0

5.4.2 实验结果

由于 BERT 模型在此数据集上没有确定的基准值，所以只能参照单标签分类的准确率。经过实验，多标签文本分类准确率为 87.86%，与单标签文本分类差距较大。原因可能有两个：第一是多标签的文本属性导致多标签多类分类比单标签分类难度更大，第二是多标签数据集中的文本长度相对较短。但是 87.86%的准确率足以满足舆情系统的需要，可以在舆情系统中进行应用。

5.5 敏感词检测

本课题在敏感词检测实验中，共检测四种类别的敏感词，分别是暴力，民生，反动，色情。检测敏感词的同时，输出整个检测过程所用的时间，用以比较不同类别的敏感词所需要的时间。

```
C:\Users\chotc\Desktop>python dfa.py
法轮功出现暴动，派私家侦探查看成人电影
**功****，派****查看***影
['色情', '暴恐', '民生', '反动']
总共耗时：0.011728286743164062s
```

图 25 敏感词检测测试

使用 DFA 算法，敏感词检测的准确率为 100%。但除了准确率，为了使该功能能够应用在实际舆情系统中，我们还需要保证敏感词检测的速度。为了测试敏感词检测功能的性能，本课题进行了两种情况下敏感词检测的实验：

（1） 敏感词长度对性能的影响

我们分别将“舆情文本多分类与敏感性检测技术研究与实现”和“舆情”加入敏感词库中，定义类别为“民生”。然后进行对比测试。本实验采用分别进行五次测试后取耗时的平均值的数据处理方法。

```
C:\Users\chotc\Desktop>python dfa.py
我的论文题目为《舆情文本多分类与敏感性检测技术研究与实现》，谢谢老师！
我的论文题目为《**********》，谢谢老师！
['民生']
总共耗时：0.010972023010253906s
```

```
C:\Users\chotc\Desktop>python dfa.py
我的论文题目为《舆情文本多分类与敏感性检测技术研究与实现》，谢谢老师！
我的论文题目为《**文本多分类与敏感性检测技术研究与实现》，谢谢老师！
['民生']
总共耗时：0.009945392608642578s
```

图 26 敏感词长度对性能的影响

根据实验结果，敏感词长度对此功能模块的性能没有太大影响。

（2） 敏感词类别数量对性能的影响

我们在原有的四种敏感词类别的基础上，对敏感词库进行删减或拆分，分别测试不同敏感词类别数量（1 种，4 种，8 种）下敏感词检测的性能。然后使用同样的原始文本进行对比测试。同样采用分别进行五次测试后取耗时的平均值的数据处理方法。

表8 敏感词类别数量对性能的影响

敏感词类别数量	总共耗时
1	0.005242109298706055s
4	0.014961481094360352s
8	0.020914793014526367s

根据实验结果，敏感词类别数量对此功能模块的性能有较大影响。原因是本实验针对每种类别的敏感词分别建立字典树，类别越多，建立字典树过程需要的时间越长。

但是本课题使用的舆情系统，敏感词类别数量固定为 4 种，而当类别数量为 4 时，

该功能模块的处理时间不会超过 0.15s。所以本课题实现的敏感词检测功能模块完全可以满足此舆情系统的需要，可以在实际应用中发挥作用。

5.6 本章小结

本章分别对短文本单标签多类分类，长文本单标签多类分类和多标签多类分类展开实验。对于短文本单标签多类分类任务，使用 TNEWS 数据集进行实验，并进行了三个参数的多次尝试与调优，得到的模型结果达到了基准值，达到了课题开题时的要求。对于长文本单标签多类分类任务，使用 THUCNEWS 数据集进行实验，对比了第三章提出的五种长文本处理方式所得到的结果，得出结论：使用文本前后截断的处理方法取得了最高的准确率，并优于所有目前主流的深度学习模型的表现。

第六章 结论和未来工作展望

6.1 结论

在文本分类任务蓬勃发展，大有用武之地的今天，实际应用更需要与时俱进，把最新最好的深度学习模型应用到文本分类以及多分类任务中，为实际应用带来提升。本课题从舆情系统出发，通过对 BERT[7]模型原理，结构和训练方法的研究，进行了针对中文文本多分类的功能需求的改进与尝试，设计了短文本和长文本两种文本分类模型，并通过实验的方法对比了几种不同的文本处理方法，根据所使用的数据集的特征选择了表现最优的一种方法应用到实践中。同时，系统集成了敏感词检测功能，进一步完善了一个舆情监测系统应当实现的功能。在保证准确率的同时，尽可能地提升处理速度，优化用户体验。在最终的系统实现上，完整地实现了三种不同场景下的业务流程，实现了一个舆情监测系统中基本的功能。本文的工作内容可以具体阐述为：

(1) 广泛学习了之前和现有的文本分类方法和技术，包括传统机器学习和深度学习方法等，并根据各种文本分类方法的优缺点选择了最合适舆情系统的 BERT 模型。

(2) 使用现有的预训练模型，利用迁移学习的优势，在充分理解舆情系统的功能需求的基础上，发挥 BERT 模型自身结构的优势，在短文本和长文本两个中文数据集上取得了良好的效果，证明了模型的可行性。同时，使用英文 BERT 模型，在英文的多标签分类数据集上实现了文本多分类模型的训练和测试。

(3) 在针对过长文本的分类问题时，本文设计了三种思路五种方法，对过长的文本进行处理，使其能够满足 BERT 模型的要求。在本文舆情系统的背景下，由于受数据集文本的总分式的文本特征影响，前后截断式的处理方式取得了最优秀的结果。但是这不代表其他的处理方法无效，应当根据不同的使用场景选择最合适的处理方式。

(4) 本文在最终的系统设计中，集成了敏感词检测功能，实现了对舆情系统中的文本的自动检测，最终实现了舆情平台上传文本的功能。通过文本分类接口，为平台上的舆情文本进行批量分类，增加标签等功能，并结合前端实现完整的舆情平台的业务流程。

6.2 未来工作展望

尽管本课题设计和训练得到的模型和算法取得了良好效果，被证明可以应用到实际的舆情系统中，但是仍然有一些不足之处可以进行有针对性的提升，需要进行后续的工作使得模型可以得到完善和提高。

首先，本实验采用的数据集为新闻类文本。众所周知，新闻类文本往往会在开头或结尾标注上文章作者的姓名或者报纸的名称，还会存在“据……报道”“作者将进一步追踪事件的发展”等无关紧要的句子。这些句子对文本分类没有作用，甚至还会干扰模型的训练。因此可以对数据集的数据进行进一步的清洗，去除文本中不必要的文字或句子，保留有用的文本，再进行预训练模型的微调过程。这样可能可以获得更好的分类效果。

其次，本实验的预训练模型是 BERT-Base 版本。如果有更好的实验条件，可以选择 BERT-Large 模型，会获得更好的表现。但是 BERT-Large 暂时还没有针对中文的预训练模型，所以在英文文本上的表现会更好。而且，对于 BERT 模型，现在已经有许多研究对它做出了不同的改进，可以利用这些改进后的新模型加以训练，提升文本分类任务的表现。

第三，对于文本摘要方法，TextRank 算法可能不是最好的解决方案。在自然语言理解领域，文本摘要与文本分类相同，也存在基于机器学习或基于深度学习的多种摘要方式。效果比较出色的如聚类[47]方法（将文章中的句子视为一个点，按照聚类的方式完成摘要），序列标注方式（建模为序列标注任务进行处理），序列标注结合 Seq2Seq[48]方法（在序列标注的基础上结合 Seq2Seq 和强化学习），句子排序方式（建模为句子排序任务）等。在未来，可以尝试这些不同的摘要方法，选择其中最适合的。此外对于文本摘要中生成式（抽象式）方法，本实验只是定性猜想其应用在 BERT 模型的文本分类任务中会对模型造成负面影响，未来可以进行必要的实验进行证明。

第四，对于多标签分类，受限于使用的数据集是英文的，所以可能模型直接迁移到中文上表现会下降。针对这一问题，可以采用网络爬虫的办法进行数据的采集，但是由于网络爬虫运行效率有限，想通过爬虫方法获得数量上万的样例数据，需要耗费大量的时间。而且爬虫得到的数据需要经过进一步的清洗和预处理，可能会存在 bad case，在未来可以进行进一步的研究。

第五，如今的网友们已经采取了很多方式绕开敏感词检测。例如，用空格隔开敏感词，或者用相同读音的别的文字代替原本的敏感词，或者用符号表达敏感词的含义等方式。如果敏感词算法可以过滤空格等间隔词，或者学习汉字中的拼音，或者赋予符号一些文字含义，就可以更加“智能”地鉴别敏感词，有效防止绕过敏感词检测的小伎俩。

最后，针对敏感词检测功能，虽然运用了状态机这样的模型，但是本质上依旧是匹配字符串的方法，在网络环境日益发展的今天，敏感词会以更快的速度出现和传播，这就要求敏感词检测功能要具有快速更新，更大容量的能力。因此如果能把机器学习甚至深度学习方法融入进来，比如上一部分介绍的 BERT 模型，利用神经网络模型学习敏感文本，获得鉴别敏感词的能力，是比字符串匹配更好的选择。但是受限于敏感词长度太短，而且敏感词的格式复杂多变，可能模型训练会有一些难度，这有待于进一步的研究。

参考文献

[1] 百度百科：文本分类[OL], https://baike.baidu.com/item/%E6%96%87%E6%9C%AC%E5%88%86%E7%B1%BB/7267115?fr=aladdinl.

[2] MARON M E, KUHNS J L. On Relevance, Probabilistic Indexing and Information Retrieval[J]. Journal of the ACM, 1960, 7(3): 216-244.

[3] PANG B, LEE L, VAITHYANATHAN S, et al. Thumbs up? Sentiment Classification using Machine Learning Techniques[C]. empirical methods in natural language processing, 2002: 79-86.

[4] YU H, LO H, HSIEH H, et al. Feature Engineering and Classifier Ensemble for KDD Cup 2010[C]. knowledge discovery and data mining, 2010.

[5] DENG L, YU D. Deep learning: methods and applications [J]. Foundations and Trends® in Signal Processing, 2014, 7(3–4): 197-387.

[6] ZHOU X, WAN X, XIAO J, et al. Attention-based LSTM Network for Cross-Lingual Sentiment Classification[C]. empirical methods in natural language processing, 2016: 247-256.

[7] DEVLIN J, CHANG M-W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding [J]. arXiv preprint arXiv:181004805, 2018,

[8] LIU X, HE P, CHEN W, et al. Multi-Task Deep Neural Networks for Natural Language Understanding[J]. arXiv: Computation and Language, 2019.

[9] YANG Z, DAI Z, YANG Y, et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding[J]. arXiv: Computation and Language, 2019.

[10] LIU Y, OTT M, GOYAL N, et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach[J]. arXiv: Computation and Language, 2019.

[11] JOSHI M, CHEN D, LIU Y, et al. SpanBERT: Improving Pre-training by Representing and Predicting Spans[J]. arXiv: Computation and Language, 2019.

[12] KIM Y, HAHN S, ZHANG B, et al. Text filtering by boosting naive Bayes classifiers[C]. international acm sigir conference on research and development in information retrieval, 2000: 168-175.

[13] POLPINIJ J, CHOTTHANOM A, SIBUNRUANG C, et al. Content-Based Text Classifiers for Pornographic Web Filtering[C]. systems, man and cybernetics, 2006: 1481-1485.

[14] MALO P, SIITARI P, WALLENIUS J. Semantic Content Filtering with Wikipedia and Ontologies. Data Mining Workshops(ICDMW)[C], 2010 IEEE International Conference, 2010,12:518-526.

[15] ZENG J, DUAN J, WU C, et al. Adaptive Topic Modeling for Detection Objectionable Text.[C]. web intelligence, 2013: 381-388.

[16] PETERS M E, NEUMANN M, IYYER M, et al. DEEP CONTEXTUALIZED WORD REPRESENTATIONS[C]. north american chapter of the association for computational linguistics, 2018: 2227-2237.

[17] ALEC RADFORD, KARTHIK NARASIMHAN, TIM SALIMANS, et al. Improving language understanding with unsupervised learning. Technical Report[R], OpenAI. 2018.

[18] WANG A, SINGH A, MICHAEL J, et al. GLUE: A Multi-Task Benchmark and Analysis

Platform for Natural Language Understanding[J]. arXiv: Computation and Language, 2018.
[19] WARSTADT A, SINGH A, BOWMAN S R, et al. Neural Network Acceptability Judgments[J]. arXiv: Computation and Language, 2018.
[20] DANIEL CER, MONA DIAB, ENEKO AGIRRE, et al. Semeval-2017 task 1: Semantic textual similarity multilingual andcrosslingual focused evaluation. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)[C], Vancouver,Canada. Association for Computational Linguistics. 2017: 1–14.
[21] LUISA BENTIVOGLI, BERNARDO MAGNINI, IDO DAGAN, et al.The fifth PASCAL recognizing textual entailment challenge. In TAC. NIST. 2009.
[22] ADINA WILLIAMS, NIKITA NANGIA, AND SAMUEL R BOWMAN. A broad-coverage challenge corpusfor sentence understanding through inference. NAACL[C]. 2018.
[23] Z. CHEN, H. ZHANG, X. ZHANG, et al. Quora question pairs[OL]. 2018. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
[24] WILLIAM B DOLAN AND CHRIS BROCKETT. Automaticallyconstructing a corpus of sentential paraphrases.In Proceedings of the Third International Workshop on Paraphrasing (IWP2005)[C]. 2005.
[25] PRANAV RAJPURKAR, JIAN ZHANG, KONSTANTIN LOPYREV, et al. Squad: 100,000+ questions for machine comprehension of text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing[C], 2016:2383–2392.
[26] LIU X, HE P, CHEN W, et al. Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding[J]. arXiv: Computation and Language, 2019.
[27] XLNet Team. A Fair Comparison Study of XLNet and BERT[OL]. https://medium.com/@xlnet.team/a-fair-comparison-study-of-xlnet-and-bert-with-large-models-5a4257f59dc0
[28] NIVEN T, KAO H. Probing Neural Network Comprehension of Natural Language Arguments[C]. meeting of the association for computational linguistics, 2019: 4658-4664.
[29] MCCOY R T, PAVLICK E, LINZEN T, et al. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference[C]. meeting of the association for computational linguistics, 2019: 3428-3448.
[30] AHO A V, CORASICK M J. Efficient string matching: an aid to bibliographic search[J]. Communications of The ACM, 1975, 18(6): 333-340.
[31] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is All you Need[C]. neural information processing systems, 2017: 5998-6008.
[32] BAHDANAU D, CHO K, BENGIO Y, et al. Neural Machine Translation by Jointly Learning to Align and Translate[J]. arXiv: Computation and Language, 2014.
[33] VINCENT P, LAROCHELLE H, BENGIO Y, et al. Extracting and composing robust features with denoising autoencoders[C]. international conference on machine learning, 2008: 1096-1103.
[34] WU Y, SCHUSTER M, CHEN Z, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation[J]. arXiv: Computation and Language, 2016.
[35] TAYLOR W L. "Cloze procedure": a new tool for measuring readability.[J]. Journalism & Mass Communication Quarterly, 1953, 30(30): 415-433.
[36] ZHU Y, KIROS R, ZEMEL R S, et al. Aligning Books and Movies: Towards Story-like

Visual Explanations by Watching Movies and Reading Books[J]. arXiv: Computer Vision and Pattern Recognition, 2015.

[37] CHELBA C, MIKOLOV T, SCHUSTER M, et al. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling[J]. arXiv: Computation and Language, 2013.

[38] PRANAV RAJPURKAR, JIAN ZHANG, KONSTANTIN LOPYREV, et al. Squad: 100,000+ questions for machine comprehension of text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing[C], 2016:2383–2392.

[39] MCCLOSKEY M, COHEN N J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem[J]. Psychology of Learning and Motivation, 1989: 109-165.

[40] THUCNEWS 中文新闻长文本标注数据集[R].http://106.13.187.75:8003/download/?dataset_name=thucnews.

[41] 今日头条中文新闻短文本分类数据集[R].http://106.13.187.75:8003/download/?dataset_name=tnews.

[42] MIHALCEA R, TARAU P. TextRank: Bringing Order into Texts[C]. empirical methods in natural language processing, 2004: 404-411.

[43] Kaggle 恶毒评论分类挑战[R].https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data.

[44] HARRIS Z S. Distributional Structure[J]. WORD, 1954, 10(2): 146-162.

[45] SUN Y, WANG S, LI Y, et al. ERNIE: Enhanced Representation through Knowledge Integration[J]. arXiv: Computation and Language, 2019.

[46] LAN Z, CHEN M, GOODMAN S, et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations[J]. arXiv: Computation and Language, 2019.

[47] JAIN A K, MURTY M N, FLYNN P J, et al. Data clustering: a review[J]. ACM Computing Surveys, 1999, 31(3): 264-323.

[48] SUTSKEVER I, VINYALS O, LE Q V, et al. Sequence to Sequence Learning with Neural Networks[J]. arXiv: Computation and Language, 2014.

谢辞

我非常感谢我的导师唐新怀教授，从最开始的选题，到难点的克服和文章框架结构的处理等，唐老师都不断给我宝贵的建议，仔细阅读我的草稿，并对此提出评论。没有他的帮助，本论文不可能达到目前的形式。

其次，我也衷心感谢所有给我讲课和大力帮助的老师，这对完成本论文是不可或缺的。我也衷心感谢帮助过我的所有同学，他们在撰写本文的过程中提供了慷慨的帮助和有益的建议,尤其是曲阜师范大学的李璐同学，她在英文文献的收集上给予我很大帮助。

最后，我想向所有在本文中提及其作品的作家表示感谢。

THE RESEARCH AND REALIZATION OF MULTI-LABEL CLASSIFICATION AND SENSITIVITY-DETECTION TECHNOLOGY OF PUBLIC VOICE TEXTS

Text classification has become an important technique for filtering out valuable and specific data from a large amount of data. Effective text classification is expected to correctly summarize the text topic, make full use of the value of data, and thus meet business needs. Text classification technology has been widely utilized in text review, ad filtering, emotional analysis and other fields.

The initial text classification relied on manual review, which was inevitably time-consuming and laborious. It then evolves to simple operation through selecting text keywords and utilizing data structures such as heaps and trees. This approach depends on an expert system and requires specialized feature rules, thus inefficient and less accurate. With the rise of artificial intelligence, machine learning methods have been used for text classification tasks. Experts use their expertise to create suitable input features, and then fit the optimal model as output by algorithm. Finally, they seek for certain features in the text, according to which the text is divided into different categories through models. Although this method increased accuracy, feature engineering and computing capabilities limit the performance of machine learning models. In order to solve the above problems, computer scientists have created a deep learning method, which enables neural networks to automatically pass through each layer to produce appropriate characteristics, so that it performs well in practical applications, covering ELMo, OpenAI GPT, MT-DNN, XLNet, RoBERTa, SpanBERT, etc. This paper studies the implementation of text classification tasks based on the BERT model, and tries different ways to optimize the results of long text classification.

BERT is a model that is trained through pre-training and fine-tuning based on the Transformer structure with the Multi-head Self-Attention mechanism. The BERT model pre-trains unmarked text represented by deep bidirection through combining the context. The embeddings of input text are categorized into three types: Token Embeddings, Segment Embeddings and Position Embeddings. Pre-training is divided into two tasks: Masked Language Model and Next Sentence Prediction. The former masks some percentage of the input tokens at random during pre-training, and then predict those masked tokens, which is designed for getting a two-way pre-training model. While the later predicts whether the next sentence is really the next sentence in order to develop the model's ability to understand sentence relationships. The pre-trained BERT model can be fine-tuned with an additional output layer to create a model appropriate for other tasks.

The experiment for this project uses a Chinese version of BERT-Base, the THUCNEWS Chinese news long text label dataset and Today’s Headline Chinese short text classification dataset. For

short text, we only need to categorize IDs and news strings. For long text, the actual sample length is up to 510 words because the BERT model has the longest sample length, 512 words, and also contains the beginning of the [CLS] and the end of the [SEP]. To solve this problem, we put forward the following proposals:

(1) Take the first 510 words of the text (head)

(2) Take the last 510 words of the text (tail)

(3) Blend the first 255 and last 255 words of the text (head and tail)

(4) Extract texts of appropriate length using text extraction method (text summary)

(5) Split text over 510 words into that of the appropriate length (split method)

The method of truncation is easily understood, but these methods will more or less lose some of the sequential information. The method (4) is the text summary method, the key point of which lies in how to filter out valid sentences and valid information. There are two text summary generation methods in natural language processing (NLP): decimation and generation. Extractive summaries often require measuring the weight of the basic sentence components and then produce the summary based on the weight results. The generated summary uses deep learning methods to interpret and abbreviate the original document, and then generates new phrases and sentences that represent the most important information in the source text. This topic chooses the TextRank algorithm of the extractive summary. It constructs the node connection diagram by dividing the text into several constituent units, with the degree of similarity between sentences as the weight of the edges. It then calculates the TextRank value of the sentence through looping iteratively. And eventually it extracts the high-ranked sentences to combine the text summary. The method (5) belongs to the text split that splits a full paragraph into several parts, each of which is less than 510 words of length.

Since we use the trained BERT pre-training model directly, we only need to fine-tune the process. The fine-tuning is divided into four phases: the training phase, the evaluation phase, the test phase and the export phase. The training phase manipulates the input text processing, the establishment of the model and the learning of the sample; the evaluation phase and the test phase are responsible for prediction, and the export phase outputs the desired format of the model.

After experiments, the accuracy of short text single label classification is up to 89.78%, reaching the current BERT-Base model benchmark value, and also the average of current mainstream text classification model. Compared with the best performing model, the accuracy discrepancy is not more than 0.05%, meeting the standard of the subject requirements which means that it can be actually applied in the real public opinion system. The head and tail method is proved state-of-the-art in the long text single label classification with a maximum accuracy of 96.02%, 0.67 percentage points higher than the baseline. This enhancement ascribes to that the head and tail method is very suitable for texts whose structure is “introduction-elucidation” , like news. Either the head or tail is proved insufficient in handling texts with a “introduction-elucidation” structure. Although the method of splitting is less accurate than the head and tail, the former retains the most complete text information, with the least loss of processing text. As for the method of text summary, due to that a definite value is relatively difficult to determine on the length of the summary (the number of sentences to be retained) the text length of the resulting summary is greatly influenced by the length of a single sentence, resulting in unstable text length and low prediction accuracy.

This topic also conducted a multi-label text classification experiment based on the English

multi-label dataset. In terms of the multi-label classification problem, each instance can be assigned to multiple categories of a set of target labels. There are two main differences between multi-label classification and single label classification: first, the label characteristics in the object is input as an array; second, we replace softmax function with sigmoid function because the latter is able to process each original output value respectively, leading to independent results. While the softmax function generates output value dependent on each other. Trained in the same way as a single-label experiment, the multi-label text classification was 87.86% of its accuracy after experiments, which was quite different from single label text classification. The difference may ascribe to two reasons: first, the property of the multi-label text brings about more difficulty than single-label classification; second, the text length of a multi-label dataset is relatively short.

Finally, in addition to the text classification, this topic also adds a sensitive word detection module, using the DFA algorithm to establish a deterministic tree finite state machine with the vocabulary in the sensitive library and querying in a circular manner. If matched, it will replace the original sensitive word with a continuous placeholder of the same length as the sensitive word and return the processed text.

On the basis of these techniques, this topic integrates the above functions on the existing public opinion systems. The main functions of the public opinion platform include the attainment of the real-time public opinion on the platform, the rapid retrieval of public opinion information text, the rapid discovery of public opinion hot spots and provision of analysis and warning, the generation of public opinion table which enables users to understand public opinion information more intuitively and comprehensively. The working process of the system can be divided into five stages: obtaining data, processing data, storing data, logical business, and user interaction. To speak more specifically, it first obtains the original data through various network crawlers. Next, the data is initially processed through Kafka, and then stored in different databases. Eventually, the business logic layer is responsible for transmitting the data in data store layers into those of a specific format that the user needs for and visually presents them to the user.

This topic encapsulates the functions of text classification and sensitive word detection at the business logic layer, realizing a complete business process of public opinion system. As a consequence, it proves the effectiveness of the text classification model and sensitive word detection module trained above.