分类号_____ 学号 M201973200

学校代码 10487 密级_____

华中科技大学

硕士学位论文

（学术型☑ 专业型☐）

基于预训练语言模型的毒性评论分类

学位申请人：李永聪

学 科 专 业：网络空间安全

指 导 教 师：鲁宏伟 教授

答 辩 日 期：2022 年 5 月 24 日

A Thesis Submitted in Partial Fulfillment of the Requirements for the Master Degree in Engineering

Toxic Comment Classification Based on Pre-trained Language Models

Candidate : LI Yongcong

Major : Cyberspace Security

Supervisor : Prof. LU Hongwei

Huazhong University of Science and Technology

Wuhan, Hubei 430074, P. R. China

May, 2022

摘 要

随着用户产生的内容 UGC（User Generated Content）的繁荣，毒性评论的传播也随之而来，这使得毒性评论检测成为一个活跃研究领域。毒性评论检测通常作为一个文本分类任务进行处理。基于预训练语言模型的方法在自然语言处理方面具有非常优异的性能，但将其应用于毒性评论分类 TCC（Toxicity Comment Classification）的研究却很少，已有的研究工作也仅将分类预测效果作为指标，而未考虑实际部署中对推断速度的性能要求。

为了更好地将基于预训练语言模型的方法应用到毒性评论分类中，对三个最流行的语言模型，即 BERT（Bidirectional Encoder Representations from Transformers）、RoBERTa（A Robustly Optimized BERT Pretraining Approach）和 XLM（Cross-lingual Language Model Pretraining），在其下游 TCC 任务上微调，设计了不同的下游网络结构，得到 TCC 任务的预训练模型的改进模型。从训练数据领域的角度去优化模型，分析了不同领域的数据分布对模型效果的影响，基于领域数据和目标任务数据，给出了领域内对模型继续预训练的策略，以及在有限计算资源下的轻量化训练参数设置。针对大规模预训练模型的部署问题，提出了一种同构 Transformer 学生模型，对预训练模型进行改进，对模型进行多阶段蒸馏，压缩部署，在性能略微损失的情况下，能够极大地提升模型在下游 TCC 任务推断速度，减少模型的参数规模，降低了部署的硬件要求。

实验结果表明，使用基础的线性下游结构优于 CNN（Convolutional Neural Network）和 BiLSTM（Bi-directional Long Short-Term Memory）等复杂的结构；用轻量超参数设置对预先训练好的语言模型进行领域内继续预训练，会给下游的毒性评论分类任务带来改进，特别是当该任务有一个相对较小的数据集时候，提升更明显；多阶段模型蒸馏的方案，在损失较少预测性能的情况下，大幅度提升的推理速度、缩小模型参数规模。

关键词： 毒性评论分类；预训练语言模型；领域内继续预训练；模型蒸馏

Abstract

With the prosperity of UGC(User Generated Content), the dissemination of toxic comments also follows. Therefore, detecting toxicity comments has become an active research field, which is usually processed as a text classification task. As a recently popular methods for text classification tasks, pre-trained LM(Language Model)-based methods are the forefront of natural language processing, and achieving state-of-the-art performance on various NLP(Neural Language Process) tasks. The method based on pre-trained language model has excellent performance in natural language processing, but few studies have applied it to TCC(Toxicity Comment Classification), and the existing research only takes the classification prediction effect as an indicator. Without considering the performance requirements for inference speed in real deployment.

To better apply the pretrained LM-based method to TCC, the three most popular language models, BERT(Bidirectional Encoder Representations from Transformers), RoBERTa(A Robustly Optimized BERT Training Approach) and XLM(Cross-lingual Language Model Pretraining), are finetuned on the downstream TCC task, with different downstream network structures added. To optimization model, the influence of data distribution in different fields on the model effect is analyzed, The strategy of continuous pre-training for the model in the domain and the corresponding lightweight training parameter Settings under limited computing resources are proposed, based on domain data and target task data. For the deployment of large-scale pre-training models, A homogeneous Transformer student model and the corresponding multi-stage distillation strategy are proposed to compress the deployment model, which can greatly improve the inference speed of the model in the downstream TCC task, reduce the parameter scale of the model, and reduce the hardware requirements of the deployment under the condition of slight performance loss.

The experimental results show that the basic linear downstream structures are superior to CNN(Convolutional Neural Network) and BiLSTM(Bi-directional Long Short-Term Memory). Continuous pretraining domain of pre-trained language models with lightweight hyperparameter Settings can improve the downstream TCC task, especially

when the task has a relatively small data set. The strategy of multi-stage model distillation can greatly improve the inference speed and reduce the model parameter scale under the condition of losing less predictive performance

Key words: TCC, pre-trained LMs, continued pre-training in domain, model distillation

目　录

1 绪论

1.1 研究背景与意义

随着各种互联网平台的迅速发展，对日常生活的影响逐步扩大，大众化用户产生的内容（User Generated Content，UGC）日益繁荣，成为内容创作的主体。但是低门槛的内容创作给网络的环境带来了巨大的隐患。其中的各种有害内容广泛而多样，以评论为代表，这些低成本产生的短文本，常常引发各种对立和极端伤害事件。人的内心往往存在黑暗的一面，而网络的匿名性诱发了人们不受现实的约束，肆意发泄评论。例如各种各样对个人的网暴事件[1]，毫无代价的随大流节奏向他人施暴，对商家产品的负面评论[2]等。轻则造成受害者抑郁，重则严重干扰了受害者的工作生活，甚至极端到心理无法承受而自杀。以及由此给互联网行业的内容安全审核也带来严峻的挑战[3]，内容安全在各项法律法规中的占比逐渐增加。即使带来巨大的人工审核成本，各大内容平台型互联网公司也不得不维持庞大的内容审核团队，来审核日益攀升的内容流量。

以社交媒体作为代表，尽管逐渐向规范方向发展，但也大量充斥着恶意的毒性文本内容，污染社区环境，引发网络节奏。在社交媒体平台上，每日会产生海量 UGC 内容，其中存在着大量未审核的不良信息，有各种形式的媒体内容，极易对青少年身心健康、社会稳定乃至国家安全[4]造成恶劣影响。因此，内容安全已成为互联网行业需要关注的重要问题。

对于不同的内容安全审核的需求，各个平台往往基于自身的数据，构建各种词典规则。结合传统的机器学习方法，通过多级标签的处理，构成业务的内容风控模块。但随着用户内容日益增长，所需庞大的数据分析、重新补充更新策略和标注数据也带来极大成本。少部分研究转向利用预训练模型去改进，而大部分平台自身的业务数据往往并未公开，且对应的规则与策略也具有定制性，难以复用。

毒性评论是存在最广泛的一种毒性内容形式，检测毒性评论成为一个活跃研究领域[5-7]。毒性评论检测通常作为一个文本分类任务处理，被称为毒性评论分类（Toxicity Comment Classification，TCC）任务。

在 TCC 研究领域，只有少数研究将预先训练好的语言模型用于 TCC，而且在探索如何最好地利用这种语言模型进行 TCC 方面的工作也很少。此外，现有的广泛使用的预训练语言模型是在书籍和新闻的正式语言上训练的，而不是在经常可以发现毒性评论的网上口语性语料上训练的。因此这也使得本文中的方法更容易被计算资源有限的用户所接受。目前尚不清楚这些现有的预训练的语言模型在多大程度上为 TCC 任务提供了实际价值。

1.2 国内外研究现状

毒性评论一般是指不同类型的用户生成的不健康和负面的内容，其中包括仇恨言论[8]、辱骂性语言[9]、网络欺凌等[10]。

毒性评论分类是一项文本分类任务。在早期的研究中，大多数的研究将 TCC 作为一个二元分类问题来处理，将一种特定类型的有毒评论与所有其他类型的评论分开[11]。后来的研究将毒性评论分类框定为多分类任务，即一个评论将被分类到多种类型的毒性评论中，或多标签任务，即一个评论可以被分配到无或至少一个（可能是多个）类型的毒性评论中。对于一个多分类任务的例子，Park[12]和 Waseem[13]等人将评论分为三类，其中每个评论属于 "无 "或 "种族主义 "或 "性别歧视"。较少有关于毒性评论分类的研究考虑多标签问题。在他们的研究中，Van 等人[14]以多分类和多标签的方式研究毒性评论分类。在他们的多标签任务中，一条评论可以被赋予 0 到 6 个标签，包括淫秽、威胁等[15]。多分类和多标签分类从本质上增加了学习难度，不同类型的毒性评论可能有一些交叉，因此增加了问题的复杂性。不过，它们更接近于现实世界的场景[16]。

传统上，文本分类任务使用统计机器学习方法，如支持向量机（Support Vector Machines，SVM），朴素贝叶斯和决策树[17]。自 2010 年以来，研究已经转向基于深度神经网络（Deep Neural Network，DNN）的模型，如卷积神经网络（Convolutional Neural Network，CNN）、循环神经网络（Recurrent Neural Networks，RNN）、双向长短记忆网络（Bi-directional Long Short-Term Memor，BiLSTM）和结合不同 DNN 配置的混合神经网络[17-19]。

在毒性评论分类这一特定场景中，各大互联网平台公司仍然将其作为稳定的解决方案。其重要原因在于，内容审核监管政策极为严格，因此也要求内容平台对自身的审核采取规则策略加以人工审核辅助的形式。基于规则词典的方式，虽然可解释性强，可控性高，但是严重依赖数据处理人员大量分析数据，并总结出规则策略，长期维护多级分类词典。在当下内容流量飞速增长更新的情况下，变得越来越难以执行。因此，目前各大内容平台的风控趋势为，通过基础模型解决大部分类别情况的毒性评论，辅助人工审核分析长尾类别，即少部分但类别较多的情况，分别处理。详细来说，简单的通过指定正则表达式的模板、或者简单的词袋模型等基于规则的方法，处理这些基础模型难以解决的情况。而基础模型部分的步骤，存在可复用的空间。好的基础模型，应当尽可能地提高检测的性能，才能更大程度减少长尾类别中的人工审核分析的工作量。

尽管各大内容平台公司积累了大量的策略，而在基础模型中，本质上是基于多种传统基于机器学习和神经网络的文本分类方法，如前面提到 TCC 通常作为一个文本分类问题，主要通过机器学习方法处理[31-33]。这些方法性能受限，往往需要上百万的标注数据量才能将性能达到勉强可用，而且需要将网络做的非常深，才能容纳足够的信息，然而由于 CNN、RNN、LSTM 这些基本结构单元本身的缺陷，将其做深会带来一系列如梯度消失、爆炸等训练问题。

自从引入 Vaswani 等人[20]提出的 Transformer 的结构后，在下游文本分类中使用预训练的语言模型已经成为主流方法。其基本思路是，从语言模型中提取预训练的神经网络层，并在此基础上增加新的神经网络层，为下游任务量身定做[25]。换句话说，来自预训练语言模型中训练好的参数包含了语言的通用知识，能够被迁移到新的神经网络中，这种训练被称为微调，后续的微调使得模型更适应为目标分类任务。由于预训练语言模型可以在无标注的数据上进行预训练，所需要的仅需大规模数据与计算资源的训练，无需标注，就能提升下游任务的性能，这极大地减少了标注数据的成本。在下游任务应用的时候也仅需要在少量标注的任务数据上进行微调，就能达到一个不错的效果。

在下游文本分类中使用预训练的语言模型，其基本过程是在预训练的语言模型

之上为下游任务添加特定的任务层，然后训练新的模型，其中只有特定的任务层被从头训练[20-22]。常用的预训练语言模型包括 BERT（Bidirectional Encoder Representations from Transformers）[20]、RoBERTa（A Robustly Optimized BERT Pretraining Approach）[22]、XLM（Cross-lingual Language Model Pretraining）[21]。这些模型在特别大的语料库上进行了预训练，例如 BERT[20]的语料库包含超过 30 亿个词。

通常情况下，预训练的语言模型中的输出层（隐藏层之后的最后的几层）会被替换成特定的任务层，然后新的模型将在目标文本分类任务上进行有监督的训练。有两种策略被广泛用于提高预训练的语言模型在下游文本分类任务中的表现：下游神经网络结构的设计[23, 24]，以及在领域内对语言模型的继续预训练[25, 26]。

（1）下游网络架构

一个用于分类任务的基本下游网络结构是一个线性分类层[20]。例如，Munikar[27]和 Mozafari[28]在 BERT 上使用线性分类层进行情感分类和仇恨言论检测；Chronopoulou[24]通过在线性分类层和预训练的语言模型之间添加一个额外的 LSTM 层来扩展基本的下游结构；Beltagy[23]在 BERT 之上增加了两层 BiLSTM。所有这些不同的网络，例如带有注意力的 LSTM、CNN，都被添加到预训练的语言模型之上，然后被送入分类层。还有许多其他研究使用类似的方法。这里不做详细介绍，请感兴趣的读者参考 Gao[29]和 Tang[30]的工作。思路上都是将预训练模型和文本分类任务传统机器学习方法[31-33]结合，然而这些结构性的改进是否具有普遍的提升的效果，在不同数据集上提升的表现能否一致，往往也没有被严格证明。另外目前关于下游神经网络结构的研究主要是基于 BERT，还没有应用到更多的最新语言模型，如 RoBERTa 和 XLM，这些是在 BERT 之后的改进工作，在某些场景下表现出更好的效果，那么进一步在此基础上增加下游神经网络结构是否能提升效果仍然值得研究。

（2）对语言模型进行继续预训练

一些研究探讨了使用大型领域内语料库对语言模型进行继续的预训练，然后将新预训练的语言模型迁移到目标分类任务上。因此，分类器模型的训练是用进一步微调的语言模型的权重初始化的。这种方法也被称为 “领域内的继续预训练”[25]。

SciBERT[23]和 BioBERT[26]是“领域内继续预训练”的两个例子，它们都使用大型领域内语料对 BERT 进行微调，并需要大量的计算资源和时间。为了解决这个问题，Gururangan[25]提出了任务自适应预训练（Task-Adaptive Pre-Training，TAPT），它使用来自下游目标任务的未标注的训练数据来进一步微调 RoBERTa。因此，在领域内进一步微调语言模型的数据规模已经大大减少。然而，在他们提出的 TAPT 中，由于超参数的设置比较高，特别是由于批量大小和训练轮次，仍然需要大量的计算资源，存在较高的门槛。

尽管 TAPT 是一种在领域内对语言模型进行继续预训练的方法，但在本文的其余部分，使用“领域内继续预训练”来指的是使用大型的无标注领域内语料库对语言模型进行继续的训练，该语料库与目标任务的无标注训练数据并不相同。另一方面，TAPT 将被用来特指更具体的使用目标任务中的无标注训练数据，在领域内继续进行预训练，即在本文中特指目标 TCC 任务的小规模数据，这里训练任务是自监督预训练任务，未用到标注。

另外，大多数研究工作仅以预测效果为目标，未考虑到往往实际应用中的 TCC 任务，缺乏对当前业界场景大规模内容信息流的考虑，这些应用大多数都需要部署成高频的服务，因此对于模型推断的速度以及参数量的大小有较高要求，对服务的速度也有较高的要求，即在提升效果的同时，也要有较快的推断速度和较小的参数规模。

1.3 本文主要内容

本文研究了如何更好地利用基于预训练语言模型（Language Model，LM）进行毒性评论分类。对比不同的下游网络结构和不同的预训练语言模型，复杂的结构是否更有效，哪种下游网络结构在毒性评论分类任务中的表现更好；在 TCC 任务领域内继续预训练是否有提升效果，进一步探索在有限资源下的改进，使用轻量级的训练参数是否也能普遍提升 TCC 任务的分类表现；分析当前实际应用部署中对模型推断性能的要求，比较了几种模型压缩方案，设计最适合 TCC 任务的方案，目的在损失极少性能的情况下提升模型的规模及推断速度。

1.4 本文组织结构

除本章外，论文其他章节的主要研究内容和组织结构如下。

第二章介绍了本文在下游 TCC 任务中改进的预训练语言模型的方法，目的是通过改进下游结构提升效果。

第三章解释了本文提出的轻量 TAPT 策略，目的是在有限资源下，更充分地利用任务数据预训练提升效果。

第四章解释了本文提出的模型压缩部署策略，目的是在损失较少性能下较大提升推断速度，减少模型规模，以便部署。

第五章介绍了本文的实验结果及评估。

第六章总结了这项工作，总结本文的创新点，并对未来的工作进行展望。

2 基于下游 TCC 任务改进的预训练语言模型

本章主要介绍实际场景中的 TCC 任务问题，模型结构和几种下游网络结构改进，在 TCC 任务上进行微调。首先对 TCC 任务进行了分析，然后介绍了模型基础部分的结构，并在下游任务 TCC 上进行直接微调，添加线性层、CNN 和 BiLSTM 三种下游网络结构后再微调，以及训练细节。

2.1 TCC 任务分析

TCC 任务本质上属于文本分类任务，是自然语言处理的经典任务，可以简单的定义为：给定文本及标签集合，将其分配标签。其中简单的毒性评论任务，往往只有正常和毒性两种标签。更复杂的毒性评论任务需要更细粒度的分类标签，这更符合实际场景中的业务需求。本文所用的数据集包含三个多类分类任务和一个多标签分类任务，与实际场景相近，方法更具有落地的价值。

一个文本分类任务过程大致包含文本预处理、文本表示和构建文本分类模型三个步骤。文本预处理是一切自然语言处理中的首要任务，其核心目的是对文本数据进行去噪、去重、去错误标签等处理，提升数据质量。具体的步骤取决于实际数据的情况、语言以及所用的分类模型种类，通常包含文本清洗、分句、分词、拼写纠正、剔除停用词等步骤。不过文本预处理步骤对于预训练语言模型之前的传统方法比较重要，并且针对特定数据集具有定制性。实际中，具体 TCC 任务往往需要对业务数据做一系列规范的处理，根据数据的来源与分布采取不同的抽样捞取策略。这些问题偏工程性，并不属于本文研究的重点，因此使用的也是公开的处理过的数据集。并且对于基于预训练语言模型的方法，由于自身本就是在通用语料库上训练的，文本预处理的方式也仅有简单的去噪等操作。本文并未对数据进行某些定制化的预处理。

文本表示是将非结构化的自然文本数据转换成词嵌入向量的形式，便于计算机运算，同时也得包含对语义的理解。其中较为传统的文本表示方法有：基于共现词频矩阵的方法（包括特征降维）、基于词频与重要性贡献的词频逆文档频率等，但

是由于过于依赖统计信息，大多数丢失了上下文语义信息及词之间的相关性，生成的词嵌入缺乏语义信息，效果受限，往往只能应用于信息检索等前置的数据挖掘的步骤。针对传统方法的局限性，基于深度学习的方法成为主流，不过以 CNN、RNN、LSTM 为基本单元的方法由于自身的缺陷，当希望将神经网络做深以容纳更多的信息时，会导致梯度消失、爆炸等训练困难的问题。后面出现的基于预训练的词嵌入模型解决了这个问题，逐渐成为主流方法，利用大规模的无标住语料库进行预训练，这些模型在不同程度上提取局部的语义信息、语法信息，从单词级别、字符级别或子词级别进行建模。

文本分类模型是在文本表示的基础上选择合适的分类算法实现分类。传统的词袋模型结合机器学习算法是经典的分类算法之一，假设文本是单词的无序集合，语序信息被忽略。传统的文本分类模型基于 CNN、决策树、RNN、LSTM 这些传统机器学习的分类结构，对经过预训练词嵌入表示的文本进行分类，往往也有不错的表现，而且速度也较快，实际应用中常常作为基线。但是当任务难度更大的时候，具体来说，类别定义难以区分，相近类别标签较多，多级标签的情况，这些传统模型的性能有限。仅使用预先准备好的通用的预训练词嵌入的文本表示，已经无法适应具体任务。只能针对任务重新定制训练对应的词嵌入表示，并更新词典。业界场景中更为通用的思路则是，既然模型规模有限，不足以容纳足够的信息，使得达到业务的要求。那么就采用规模更大的预训练模型，更大的隐层维度表示，更多的层数，继续训练达到更高的上限。以及数据侧针对性的补充迭代，由于各大内容平台公司积累了大量的数据，在实际业务场景中数据获取更为容易，按照数据>特征>模型的优先级去迭代。不过本文的研究的重点虽然侧重于实际场景，但是不依赖在数据、特征前两个维度的工作，而是关注模型维度的方法，这更具有可复用价值。

大量的改进工作试图将预训练模型与传统的 CNN、RNN、LSTM 等神经网络结构融合，并在某些数据集上取得了更好的效果。但是往往不稳定，这些结构是加在基础预训练模型上的，参数自然也只能随机初始化，在微调训练的过程中收敛。评估这些结构改进的作用，及什么场景下哪种结构起到优化的作用值得研究。直觉上，由于这些下游结构并未参与预训练，起到的作用应该有限。

2.2 TCC 任务微调及下游网络结构改进

本节将对 BERT、RoBERTa、XLM 三种预训练语言模型在下游 TCC 任务上进行微调，采取直接以开头的[CLS]作为整句文本的表示，以及将线性层、CNN、BiLSTM 三种下游网络结构拼接在模型整句的输出后作为整句文本的表示，下游 TCC 的多类和多标签分类任务。

2.2.1 模型结构

本节以 BERT、RoBERTa、XLM 三种预训练语言模型作为基础部分，添加线性层、CNN、BiLSTM 三种下游网络结构，整体结构如图 2-1 所示。

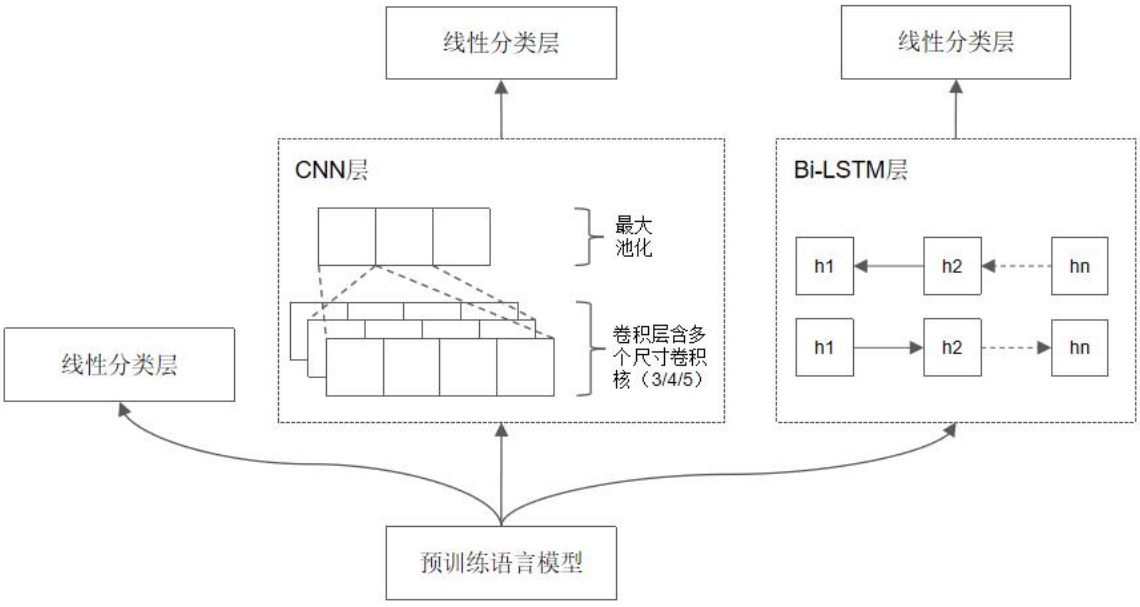

图 2-1 整体结构

在当前的文本分类任务中，基于 Transformer 结构的预训练语言模型是主流方法。能够在大规模的通用无标注语料库上进行预训练，获取文本通用的语义语法知识，提升迁移到具体任务数据上的效果。这种训练任务被称为自监督预训练任务，即利用自然文本数据自身结构的信息学习，保存在模型的参数中。一般来说，浅层表示偏向通用的语法信息，深层表示偏向特定任务的语义信息。其中代表模型是 BERT，由此开启了自然语言处理的预训练微调范式的时代，使得实际各种场景的下游任务

得以落地，仅需要目标较少的标注数据，就能达到一个基本可用的效果。以及之后对 BERT 各种维度层出不穷的改进工作，至今为止较为流行的两个改进工作为 RoBERTa 和 XLM，它们的结构完全相同，RoBERTa 主要在预训练任务数据的处理及训练方式上改进获得通用性能提升，而 XLM 主要在多语言场景针对性数据处理及预训练任务设计上改进获得特定场景下的性能提升。这三种是实际中应用最多的模型，具有代表性，总体区别主要在于预训练任务以及词嵌入的处理。

Transformer 是组成以 BERT 为代表的预训练语言模型的核心模块，其中 Attention 机制是 Transformer 中最关键的组成部分。是本节提出的改进模型基础结构部分，以及后面第 4 章提出学生模型的结构中的基本单元，下面依次介绍从最小单元到整体。

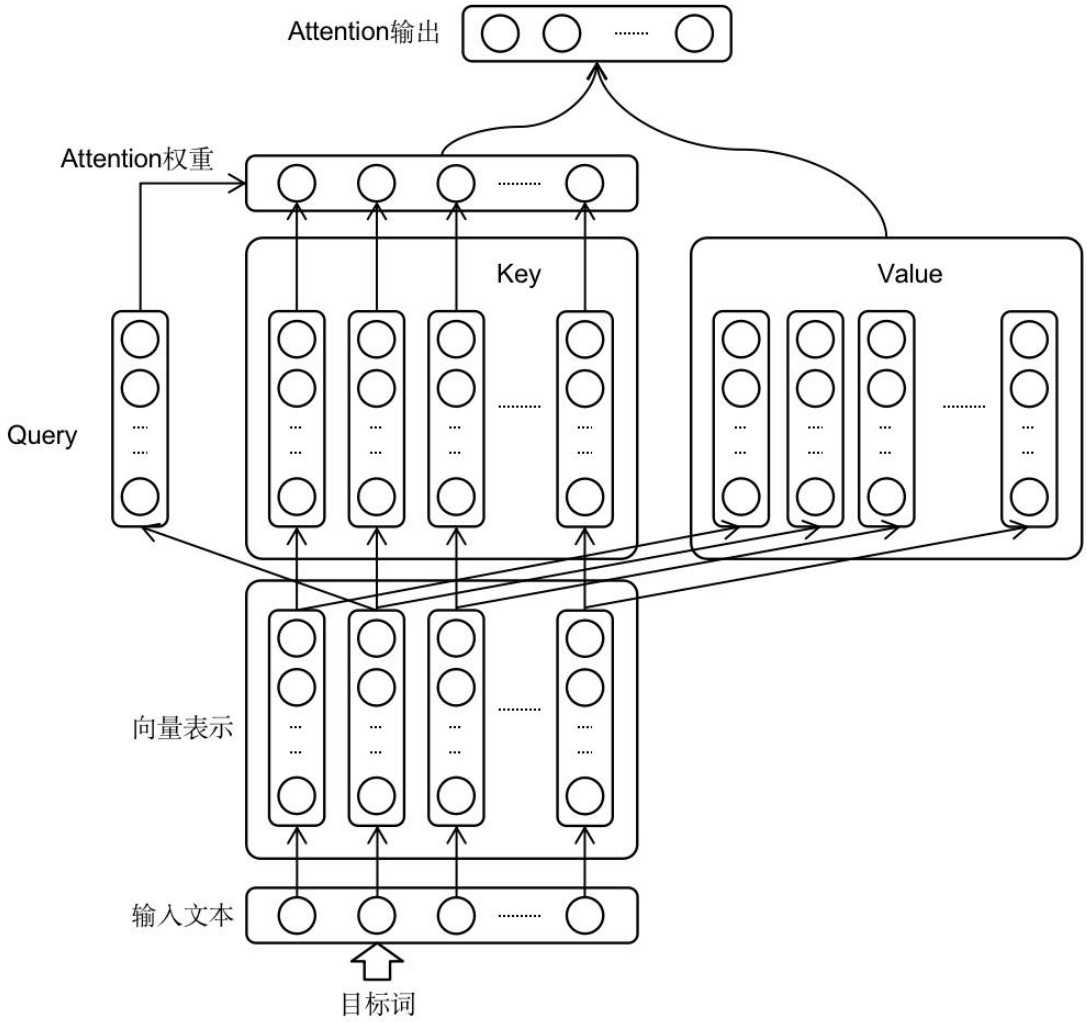

图 2-2 Attention 机制

（1）Attention 机制

也称为注意力机制，主要作用是让神经网络将注意力放在输入序列的一部分位置上。Attention 机制借鉴了信息检索中的 Query、Key、Value 概念。如图 2-2 所示，Attention 机制将目标词向量和上下文的各个位置的词向量进行计算，作为当前位置词的表示向量。首先通过三个随机初始化的权重矩阵 W_Q、W_K、W_V 线性变换，获得目标词的 Query 向量表示、上下文各个词的 Key 向量表示和目标词及上下文各个词的原始 Value 向量表示，然后计算目标词的 Query 向量和各个位置词的 Key 向量的相似度作为 Attention 权重，即注意力放在不同的位置不同，加权求和目标词的 Value 向量和各个位置词的 Value 向量，作为 Attention 的输出，本质上就是用词的上下文的信息增强自身的语义向量表示。

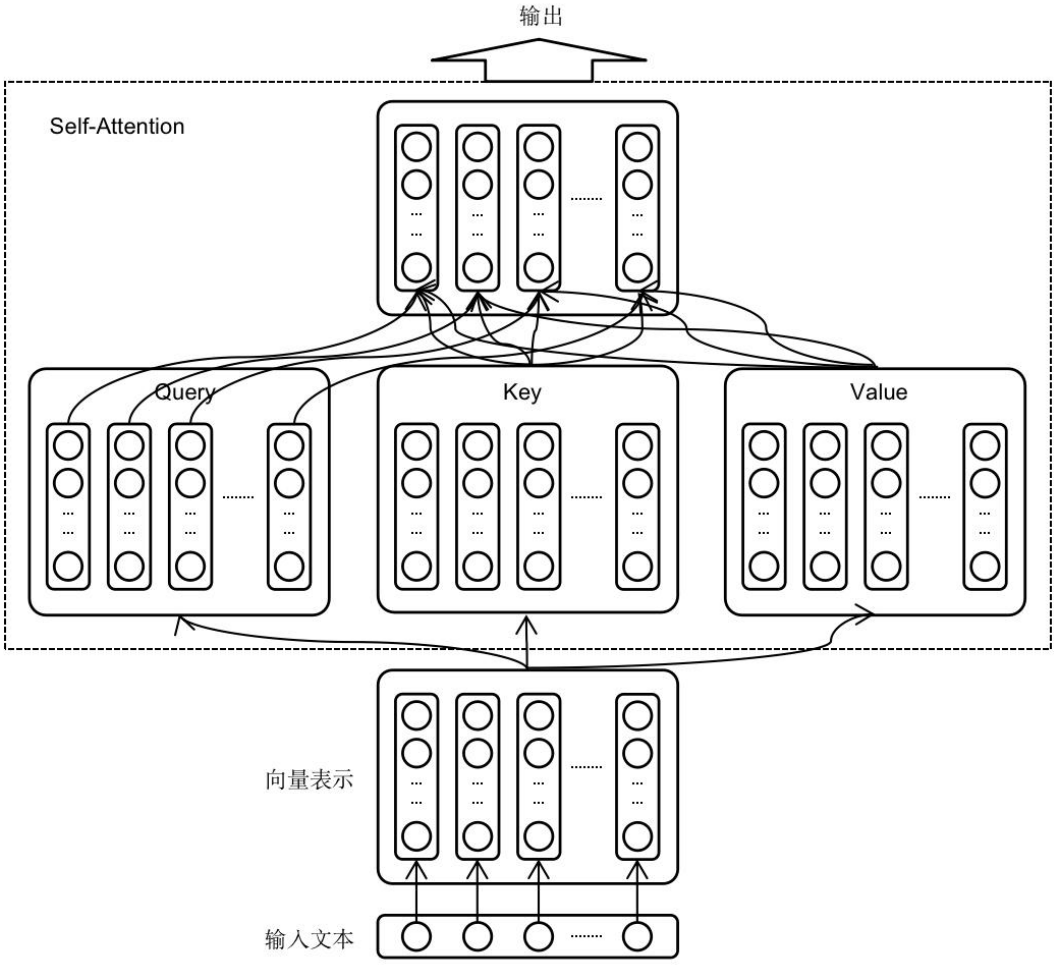

图 2-3 Self-Attention 结构

（2）Self-Attention

对于输入文本序列的每个位置的词，都需要增强语义向量表示。同样的，对每个词进行 Attention 计算，加权计算文本中所位置词的向量，得到各个词的 Attention 输出，即增强语义向量表示。

如图 2-3 所示，这种情况下，Query、Key、Value 的向量均来自同一文本序列，因此被称为 Self-Attention。如公式（2.1）所示，由 W_Q、W_K、W_V三个权重矩阵线性变换生成输入序列对应的Q、K、V矩阵，d_k为向量维度，这里的 W_Q、W_K、W_V三个权重矩阵是随机初始化后在训练过程中学习。

$$\text{Attention}(Q,K,V)=\text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

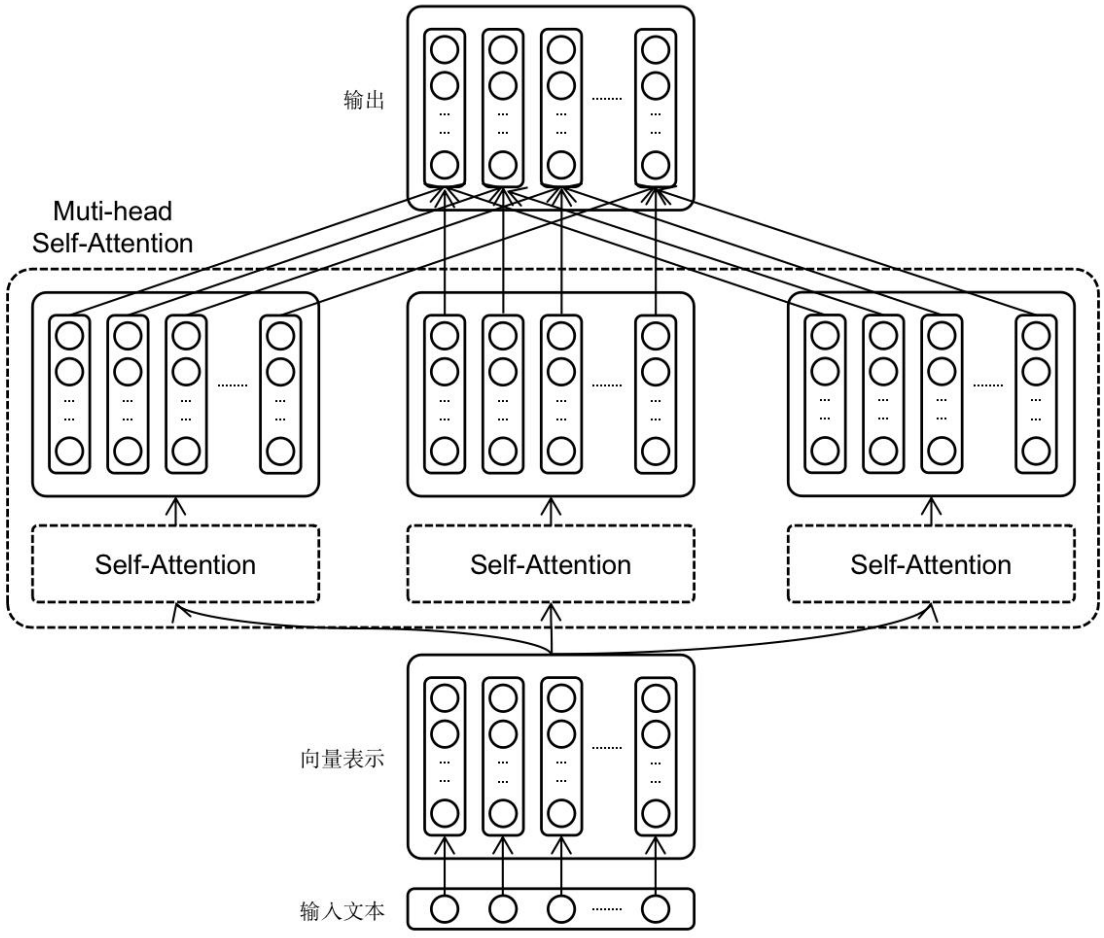

图 2-4 Multi-head Self-Attention 结构

（3）Multi-head Self-Attention

为了增强丰富 Attention 的多样性，利用不同的 Self-Attention 模块，及对应的不同的 W_Q、W_K、W_V 权重矩阵，进一步获得每个词在不同的语义空间下的增强语义向量表示，然后将多个 Self-Attention 模块的输出进行线性拼接和变换，得到一个与原始词向量相同维度的增强语义向量。不过这种设计存在一定的冗余，在第 4 章中的压缩方法就有针对其冗余结构的裁剪。如图 2-4 所示，可以理解为考虑多种语境下目标词与上下文其它词的语义向量不同的表示。思想上与传统的 CNN 多组卷积核得到不同的特征空间表示相同。

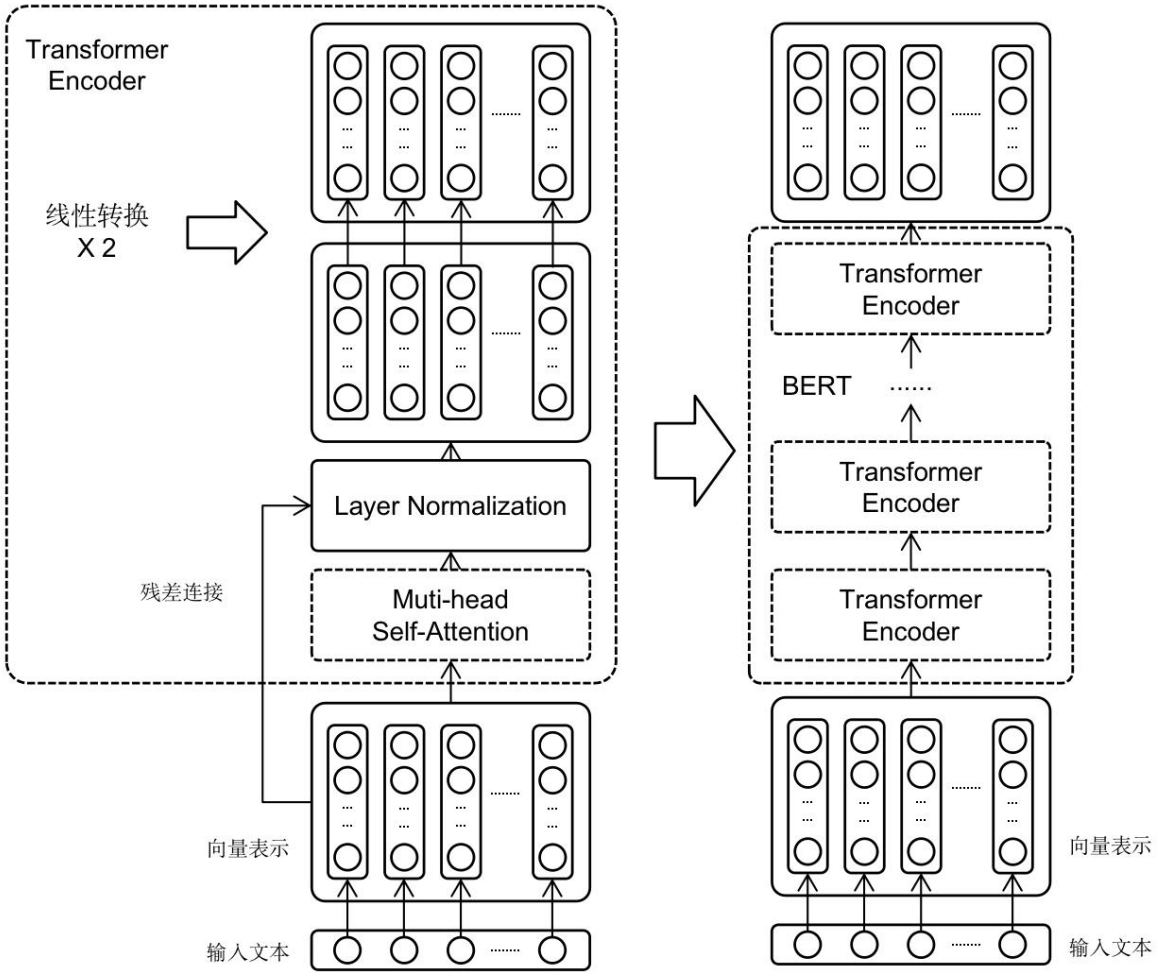

图 2-5 Transformer Encoder 堆叠

（4）Transformer Encoder

在 Multi-head Self-Attention 的基础上再添加一些结构，就组成了 Transformer Encoder。如图 2-5 所示，左侧为 Transformer Encoder 结构，在 Multi-head Self-Attention

之上加了以下三个关键操作：残差连接，将输入直接加到输出，这使得模型更容易叠加深度，而在训练中减缓梯度消失；Layer Normalization，对输出作层标准化，以稳定梯度分布，避免后面进过激活函数的时候进入梯度平缓区间，同样减缓了梯度爆炸和消失的情况；线性转换，一般为一个前馈神经网络，进行两次线性变换，以增强表示能力，变换后的维度与原来保持一致。如图 2-5 的右侧为多个 Transformer Encoder 层层堆叠，构成了 BERT 模型。

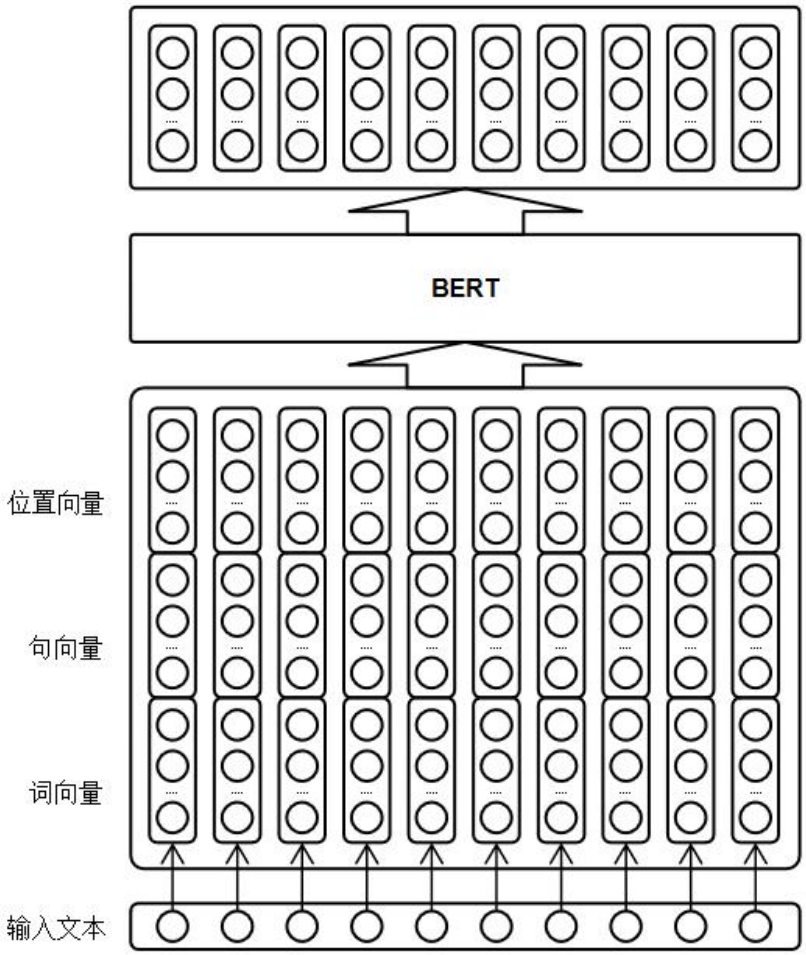

图 2-6 输入文本嵌入

（5）输入文本嵌入

在基于深度神经网络的各种 NLP 方法中，通常使用一维向量来表示。每个位置以文本中的每个最小单元为单位，其中以字为最小单元的，一般称为字嵌入向量；以词为最小单元，一般称为词嵌入向量。注意这里的词可以是各种分词方法得到的

分词、或子词。如图 2-6 所示，词嵌入将自然文本句子表示成词向量、句向量、位置向量相加作为输入向量。

BERT 代表的含义是 Transformers 的双向编码器表示，与先前的一些语言表示模型不同，BERT 旨在通过对所有层的左右上下文进行联合调节，从无标注的通用文本数据中预训练文本的深度双向表示。概率建模的目标函数如公式（2.2）所示，w_i为每个位置的单词。因此，只需一个额外的输出层就可以对预训练的 BERT 模型进行微调，从而为广泛的各种下游 NLP 任务，如问题回答和语言推理，构建最先进的模型，而无需对特定任务进行大量模型结构的修改。

$$P(w_i|w_1,\ldots,w_{i-1},w_{i+1},\ldots,w_n) \tag{2.2}$$

在本文的 TCC 任务中，所用的三种预训练语言模型，基础结构部分都是和 BERT 相同的，区别主要在于输入的处理以及预训练任务，而在第 3 章中才涉及自监督预训练，本节中的微调训练是监督训练过程，第 4 章中的蒸馏训练也涉及到两种训练任务。

2.2.2 模型的输入输出

在文本预处理中，基于预训练语言模型的方法往往都有其对应的将文本嵌入转化成模型所需要的输入向量形式处理，被称为 Tokenizer。对 BERT 而言，将英文词进一步切割，使用 WordPiece[38]词嵌入划分为更细粒度的语义单位。在此基础上，深度神经网络将整句转化成一维向量，经过各层复杂的计算转换后，输出一个一维的句向量，作为文本的语义表示。好的文本表示通常会希望语义相近的词在特征向量空间上的距离比较接近，这样一来，通过词向量转换后的文本句子向量才能表示更加准确的语义信息。如图 2-6 所示，模型输入，首先将文本中各个词转换成原始的词向量，既可以采取随机初始化，也可以采取其它词向量 Word2Vector[39]等算法的词表作为初始值；输出是文本中各个词融合来上下文语义信息后的向量表示。另外，从图中还可以看出，模型的输入，除了词向量外，还包含以下两个部分。

（1）句向量

该向量主要用于模型的预训练任务下一句预测任务中表示输入的两句是否连续，分别用 0 和 1 表示，当前位置的词属于前一句还是后一句。

（2）位置向量

由于 Transformer 结构抛弃了 RNN 的循环结构，词的位置信息也被抛弃，而实际出现在文本中的不同位置的词所携带的语义信息又依赖位置信息，所以 BERT 对此使用位置向量来补充缺失的位置信息，使用绝对位置嵌入的方式。

最后，将词向量、句向量、和位置向量加和作为模型的输入。对不同的 NLP 下游任务，模型的输入会有微调，对模型输出利用也存在一些差异。单句文本分类任务：对于文本分类任务，BERT 模型在句子前插入一个特殊的[CLS]Token，并将该 Token 对应的输出向量作为整句文本的语义表示，用于文本分类任务。可以理解为，在 Transformer 结构层层计算各个词注意力的过程中，融合整句所有词的信息，这个无明显语义信息的符号能够更加“公平”的融合文本中每个词的语义信息。句对分类任务：该任务的实际应用场景下游任务包含问答（判断一个问题与一个答案是否匹配）、语句匹配（两句话是否表达相同的语义）等。对该任务，BERT 模型除了添加[CLS]，并将其对应的输出作为文本的语义表示，还对输入的两个句子用一个特殊的[SEP]Token 分割开来，并分别对两句添加两个不同的句向量作为区分。序列标注任务：该任务的实际应用场景下游任务包含：命名实体识别（标注每个词的词性）、中文分词（标注每个字是词的首个字、中间字还是末尾字）、阅读理解（标注答案的起止位置）等。对该任务使用句子中的每个位置的输出向量对该字的词性进行分类。

2.2.3 模型的预训练任务

BERT 实际上是一个语言模型，通过在大规模无标注的文本语料上自监督训练，其目的是为了学习语言自身的一个好的表示，在语法、语义上与自然文本分布的一致。其预训练的过程就是逐渐调整模型参数，使得能够很好的表示语言，便于在后面的下游任务中作微调。为了达到这个目的，提出了代表性的两个预训练任务：掩盖语言模型（Masked Language Model，MLM）和下一句预测（Next Sentence Predict，NSP）。

MLM 的任务描述为：给定一句话，随机抹去这句话中的一个或几个词，要求根据剩余词汇预测被抹去的几个词分别是什么，类似完形填空完形任务[40]，依赖上下

文信息去预测词。具体做法是：随机选择输入文本序列中的 Token（根据所用的分词算法，分成子词后的最小单元被称为一个 Token），并用特殊的[MASK]Token 替换。MLM 的目标是预测被掩盖的 Token 的交叉熵损失。BERT 采取统一 15%的概率对 Token 进行随机替换。选择的 Token 当中，80%用[MASK]Token 进行替换，10%保持不变，10%被随机的 Token 所替代。在原始实现中，随机掩盖和替换在开始时执行一次，并在训练期间保存，尽管在实践中，数据被复制，因此对于每个训练句子的掩盖部分并不总是相同的。

NSP 的任务描述为：给定一篇文章中的两句话，判断第二句话在文本中是否紧跟在第一句话之后，是一个二分类任务，正例是通过从文本语料库中提取连续的句子来创建的，负例是通过将不同文档的片段配对而创建的。让模型能够学习语义连贯性表示，正例和负例的抽样概率相等。

BERT 通过对 MLM 和 NSP 进行多任务学习，使得模型输出的每个词向量都能尽可能全面、准确且符合语法语义地表示输入文本的整体信息，为下游任务的微调提供更好的模型参数初始值。

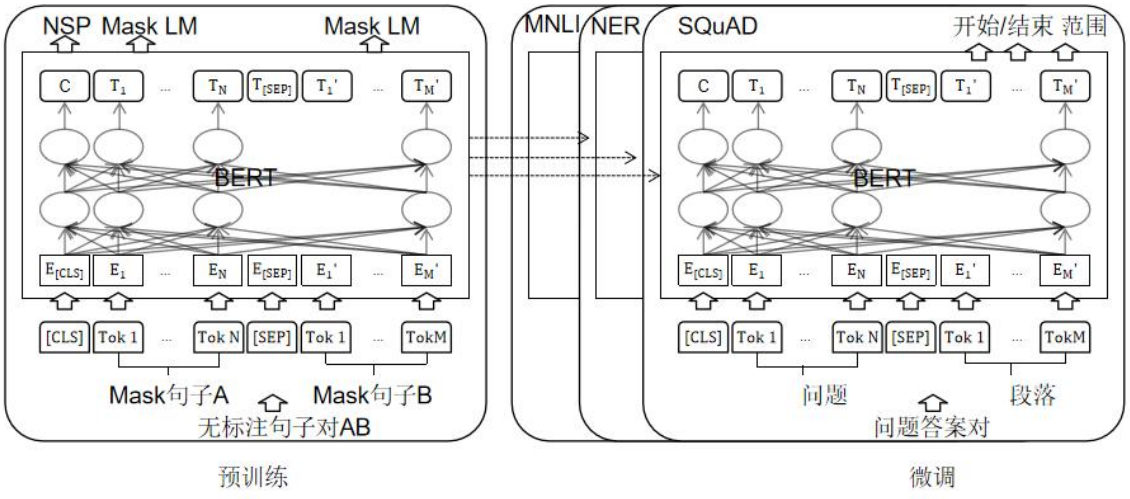

图 2-7 BERT 的整体预训练和微调过程

如图 2-7 所示，在 BERT 的框架中有两个步骤：预训练和微调，在预训练期间，模型在不同的预训练任务中使用无标注的数据进行训练。

对于微调，首先用预先训练的参数来初始化 BERT 模型，然后使用来自下游任务的标记数据来微调所有参数。每个下游任务都有单独的微调模型，即使它们是用

相同的预训练参数进行初始化的，以图中的问答任务作为示例。除了输出层，在预训练和微调中都使用相同的结构。对于不同的下游任务，使用相同的预训练模型参数来初始化模型。在微调过程中，所有参数都会进行微调。[CLS]是添加在每个输入示例前面的特殊符号，而[SEP]是特殊的分隔符（例如，分隔问题/答案）。

BERT 的一个显著特点是其跨不同任务的统一架构。预先训练的模型结构和最终的下游模型结构之间的差异很小。对于给定的 Token，其输入表示是通过对相应的 Token 嵌入、句嵌入和位置嵌入求和来构造的。

从图 2-8 中看到，BERT 的 Tokenizer 具体过程，不同的语言模型往往有不同的处理方式，在 Token 嵌入上添加的其他嵌入向量原因是服务其特定的预训练任务。

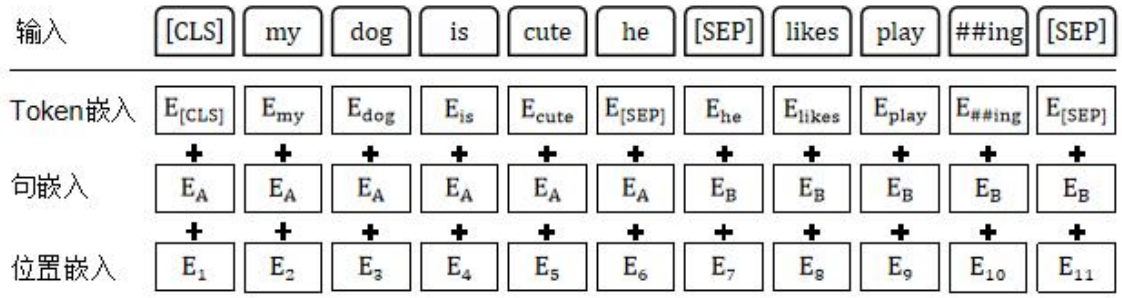

图 2-8 BERT 输入表示

预训练的数据决定了学到的通用语言表示的上限，训练前的数据准备在很大程度上遵循了关于语言模型训练的现有文献。对于预训练语料库，使用 BOOKCORPUS（8 亿词）[41]和英文维基百科（25 亿词）。对于维基百科，只提取文本段落，而忽略列表、表格和标题。关键是使用文档级语料库，而不是洗牌过的句子级语料库，如十亿词的基准[42]，以提取长的连续序列。

BERT 的微调很简单，因为 Transformer 中的自注意力机制允许 BERT 通过交换适当的输入和输出来对许多下游任务进行建模——无论它们涉及单个文本还是文本对。对于涉及文本对的应用，一种常见的模式是在应用双向交叉注意之前独立编码文本对，例如 Parikh 等人[43]和 Seo 等人[44]。相反，BERT 使用自注意力机制来统一这两个阶段，因为编码具有自注意力的串联文本对有效地包括两个句子之间的双向交叉注意。对于每个任务，只需将特定于任务的输入和输出插入到 BERT 中，并端到端地调整所有参数。在输入时，来自预训练的句子 A 和 B 类似于释义中的句子对、

蕴涵中的假设——前提对、问题回答中的问题——段落对、文本分类或序列标注中的简并的文本——标注对。在输出端，Token 的输出表示被送到 Token 级任务的输出层，例如序列标注或问题回答，并且[CLS]的输出表示被送到输出层用于分类，例如蕴涵或情感分析。与前期预训练相比，微调成本相对较低。

语言模型预训练带来了显着的性能提升，但不同方法之间的仔细比较具有挑战性。训练的计算成本很高，通常在不同大小的私有数据集上完成，超参数选择对最终结果有重大影响。

RoBERTa 主要工作是在 BERT 基础上做了几点调整，BERT 训练严重不足，使用更多的训练数据，更大的批次大小，在更长时间的轮次进行训练；移除了 NSP、训练序列更长；调整 BERT 在 MLM 任务中采取的静态掩码机制，采取更高效训练的动态掩码机制，获取了更好的性能。

XLM 则是引入了多语种的语料，以及翻译语言模型的预训练任务，在多语种的场景下获得更好的表现。XLM 提出了一种新的基于跨语言建模的无监督跨语言表征学习方法，并研究了两个单语种预训练目标。引入了一个新的监督学习目标，在有并行数据的情况下，改进了跨语言预训练。在跨语言文本分类、无监督机器翻译和有监督机器翻译方面的表现明显优于以往的水平。并且跨语言语言模型可以显著改善低资源语言的困惑性。

相较于 BERT，XLM 通过 Sennrich 等人[49]提出的字节对编码 BPE（Byte-Pair Encoding）的子词分词算法，如 Lample 等人[50]所提出对所有语言创建一张相同共享词汇表。这极大地改进了在共享相同字母表或特定的 Token（如数字[51]或专有名词）的语言之间嵌入向量空格的对齐，即在不同语种中，某一些相同形式的表达。从单语语料库中随机抽样的句子拼接构成训练数据，训练 BPE 分词器。句子按照多项分布概率$\{q_i\}_{i=1\ldots N}$进行抽样，这里$q_i = \frac{p_i^\alpha}{\sum_{j=1}^{N} p_j^\alpha}$，其中$p_i = \frac{n_i}{\sum_{k=1}^{N} n_k}$，XLM 考虑取$\alpha$=0.5。对该分布进行采样会增加与低资源语言相关联的 Token 数量，并减轻对高资源语言的偏向。特别是，这防止低资源语言的单词在字符级别被拆分，否则就会造成一个完整语义的单词被切分成多个无意义的字符片段，这不利于语义的表示。

XLM 提出了几个新的预训练任务。

（1）因果语言模型 CLM（Causal Language Modeling）

由一个 Transformer 语言模型组成，建模为如公式（2.3）的概率，对给定句子中前面的单词，预测后面单词的概率，w_t表示第 t 个词，θ表示条件因子。

$$P(w_t|w_1,\dots w_{t-1},\theta) \tag{2.3}$$

虽然 RNN 在语言建模基准上获得了最先进的性能[51]，但 Transformer 结构的模型也非常有竞争力[53]。在使用 LSTM 建模语言模型任务的情况下，通过向 LSTM 提供先前迭代的最后隐藏状态来执行通过时间的反向传播[57]。而在使用 Transformers 模型的情况下，可以将先前的隐藏状态传递给当前批次[55]，以便为批次中的第一个单词提供上下文。然而，这种方法并不能简单扩展到跨语言的设置中，所以为了简单起见，XLM 只保留每批中的第一个单词，而不是上下文。

（2）掩盖语言模型 MLM（Masked Language Modeling）

XLM 仍然考虑了 BERT 的掩盖语言模型 MLM 目标，也被称为完形任务[40]。按照 Devlin 等人[20]的研究，XLM 从文本流中随机抽样 15%的 BPE 标记，80%的时间用[MASK]Token 替换，10%的时间用随机 Token 替换，10%的时间保持不变。但是，XLM 的方法和 BERT 的 MLM 的差异，包括使用任意 Token 数量长度的句子（按 256 个长度的 Token 处为一句截断）的文本流，而不是句子对。为了应对罕见的 Token 和频繁的 Token 之间的不平衡（例如标点符号或停用词），XLM 还使用类似于 Mikolov 等人[45]的方法进行子采样：文本流中的 Token 根据多项式分布进行采样，其权重与其频率倒数的平方根成正比。MLM 的任务目标类似于 BERT 的掩盖语言建模 MLM 目标。但是区别在于，使用连续的文本流，而不是句子对。

（3）翻译语言模型 TLM（Translation Language Modeling）

CLM 和 MLM 的任务目标都是无监督的，只需要单一语言的数据，即自监督训练。然而，当多语种的并列数据可用时，这些任务目标便不能有效利用这些数据。XLM 引入了一种新的翻译语言模型 TLM 任务来改进跨语言预训练。翻译语言模型 TLM 任务目标是 MLM 的扩展，其中他们不只是考虑单一的语言文本流，而是连接并列句子。翻译语言模型 TLM 任务随机掩盖源语言句子和目标语言句子中的单词。为了预测一个在英语句子中被掩盖的单词，该模型既可以关注周围的英语单词，也

可以关注法语翻译，从而通过注意力的计算使模型对齐英语和法语表示，从而预测一种语言的单词的时候不仅使用了自身语言句子的信息，还用到了同语义另外语言的句子中单词的信息。特别是，如果英语模型不足以推断被掩盖的英语单词，则该模型可以利用法语句子推断上下文。

2.2.4 直接在 TCC 任务微调

本文基于预训练语言模型，在下游 TCC 任务上微调，并添加了三种下游网络结构作为改进。

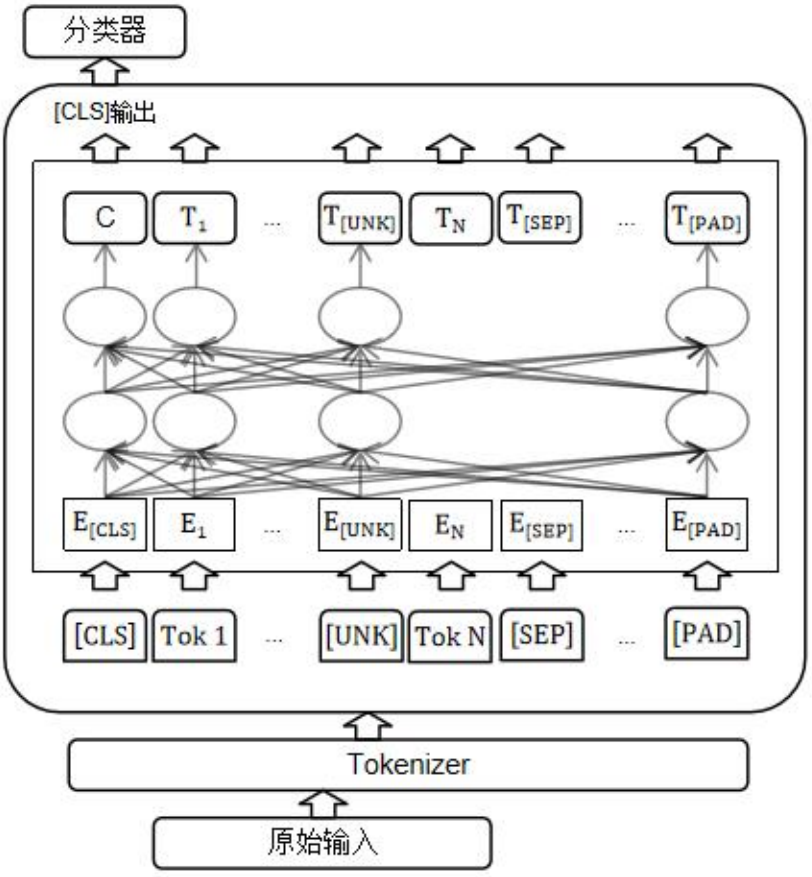

图 2-9 直接使用[CLS]输出进行多类分类

首先介绍直接微调的具体步骤。如图 2-9 所示，对于 BERT，遵循原始 BERT 的建议，直接使用开头的[CLS]Token 的最后一层输出，作为整句文本的表示，输入到 softmax 激活函数，输出类别的概率分布，如公式（2.4）所示，C是最后一层隐藏的输出，W是权重。

$$P = \text{softmax}(CW^T) \tag{2.4}$$

注意到数据的输入输出流，原始输入经过 Tokenizer，即分词器，将原始输入的自然语言文本流处理成嵌入向量的形式。该过程为：将句子切分成 Token，并添加几个特殊的 Token：[CLS]，前面已经提过，[CLS]对应的输出向量是添加在句首的，在注意力计算的过程，公平的融合各个位置的语义信息，作为整句文本的语义表示，可直接用于文本分类；[SEP]，该 Token 用于分割句子，一般用于输入为句对的情况，这里未用到，而只将其拼在句尾作为结束判断；[UNK]，即未知词 OOV(out of vocabulary)，表明未被词表识别，还是采用字符级别的 BPE 编码生成的词表，包含 3 万左右的单词，如果数据中的这类 OOV 词会大大影响模型的性能；[PAD]，BERT 模型的输入有最大长度的参数配置，因此当句子长度小于模型输入的最大长度时，需要对齐到最大长度，其对应的向量为 0，在每层计算注意力的时候会根据 0 的位置生成一个三角矩阵，将对应位置的注意力权重置 0，不会引入额外信息，而当输入句子大于最大长度的时候，则需要对句子按最大长度截断，超出的长度被截断丢弃。如图 2-10 的 Tokenizer 过程所示，将两个句子切分并添加特殊 Token，只有四个词被识别。

```
["Hello, y'all!", "How are you 😁 ?"]
["[CLS]", "How", "are", "you", "[UNK]", "?", "[SEP]", "[PAD]"]
```

图 2-10Tokenizer 切分并添加特殊 Token

将输入嵌入成向量表示，如前面的图 2-8 所示，将输入文本进行表示成 Token，然后通过词表映射成对应的向量，词表取决于 BERT 的原始预训练语料库，向量维度即每层 Transformer 的隐层输出，按照 BERT 为 768 维，维度越大，对信息的表示能力也就越强。根据 Token 的位置，生成一个句嵌入向量，对于输入为句对的情况，用于区分前后句子，一般前句为0，后句为1。同样这里未用到，全部置为1。Transformer 结构抛弃了 RNN 的前后位置信息，而对应语义表示，这也包含了语义信息，因此需要产生一个位置嵌入向量，来补充位置信息。然后将 Token 嵌入 E_{Token}、句嵌入 $E_{句}$、位置嵌入 $E_{位置}$的向量相加得到输入 E。计算如公式(2.5)所示：

$$E = E_{Token} + E_{句} + E_{位置} \tag{2.5}$$

然后经过前向传播过程，在最后一层的输出中取[CLS]位置的向量作为整句的表示。其中对于多类分类任务，预测只能是某一个类别，将输出向量映射为类别数维度的向量，经过 softmax 函数激活，获得每类的概率分布；对于多标签分类任务，预测可能为多个类别，因此也是将输出向量映射为类别数量的向量，但是经过 sigmoid 函数激活，对那些大于 0.5 阈值的维度对应的类别，预测为包含这些类别。

而对于 RoBERTa 模型，由于采用的是 byte 级别的 BPE 编码，使用里更大的预训练语料库，词表更大，OOV 问题对于模型的影响也更小，使用官方词表，包含 5 万左右词。另外一个区别在于，由于 RoBERTa 移除了 NSP 任务，所以在原始输入的时候，没有句嵌入。微调其它地方的处理和 BERT 一致。

对 XLM 模型，为了获得跨语言信息，所有的语种共用一个词表，通过 BPE 生成，共享的内容包括相同的字母、前后缀、Token、专有名词等。这种共享词表能够显著提升不同语言嵌入表示的对齐效果。XLM 同样也移除了 NSP 任务，另外还引入了新的 CLM、TLM 任务，其中为了获得跨语言表示而进行的 TLM 任务是有监督的，需要额外的向量嵌入标识句子中各位置的词属于什么语种，多了一个语言嵌入。

2.2.5 添加下游网络结构改进后微调

对于用[CLS]位置的输出作为整句的语义表示，直接用于下游分类任务。本文提出疑问，[CLS]是 Tokenizer 拼接到原句子句首的特殊 Token，没有通过词表映射，仅通过多层 Transformer 的注意力计算过程中与其它位置的词进行运算，才获得整句的语义表示，然而维度也是和其它位置一样的维度嵌入。而对于传统的文本分类方法，存在一个共识，即用一个维度较小的向量表示更大位置范围的多个向量时，一定会损失。

按照传统的分类方法，一般是通过增加线性层、CNN、BiLSTM 神经网络结构，对特征信息进行提取。前面提到，在文本分类任务，有大量工作在预训练模型基础上添加这些传统的神经网络结构，来改进在具体目标任务中的表现，得出促进了预训练模型的结论。由于大多是在各自的数据集上实验，而在实际应用中，这类改进下游结构的方法往往不稳定，有时效果甚至还不如不改，即使最终提升了模型在任务中的效果，也难以下结论是这些添加结构起到的作用。

为此，为了验证两者的效果，将预训练语言模型作为特征提取器，采取传统文本分类方法的思路，在后面添加线性层、CNN、BiLSTM 三种下游结构改进。

（1）添加线性层，如图 2-11 所示。

因为需要和直接用[CLS]对比，简单用一层线性层神经网络，将不包含句首[CLS]位置的输出映射为类别维度向量。区别仅在于前者按照 BERT 原论文建议，将[CLS]位置的输出作为整句的语义表示，后者不使用[CLS]位置的输出，而使用其它每个位置的输出经过线性层作为整句的语义表示。

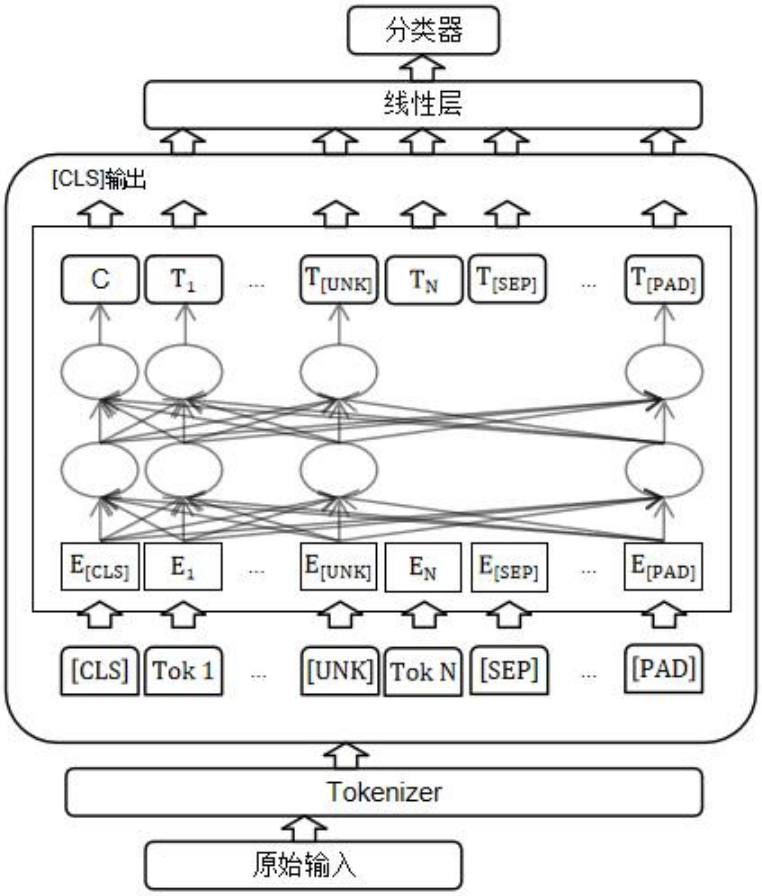

图 2-11 添加线性层结构

（2）接着添加 CNN 层和线性层，如图 2-12 所示。

第二种结构建立在第一种结构之上，但在线性层和语言模型之间有卷积层。语言模型的各层的输出（不包括语言模型头）被送入一个 3 层的卷积网络。这种 CNN 配置与 Kim[53]提出的配置相同，对文本只有一维方向上做卷积操作，其中包括三个尺

寸的卷积核，分别为 3、4 和 5。每种尺寸卷积核的输出有 100 个特征图，这也是传统的冗余特征空间表示的思路。每个卷积层的输出被传递到一个最大池化层，然后拼接在一起，再输入到最后的线性分类层。

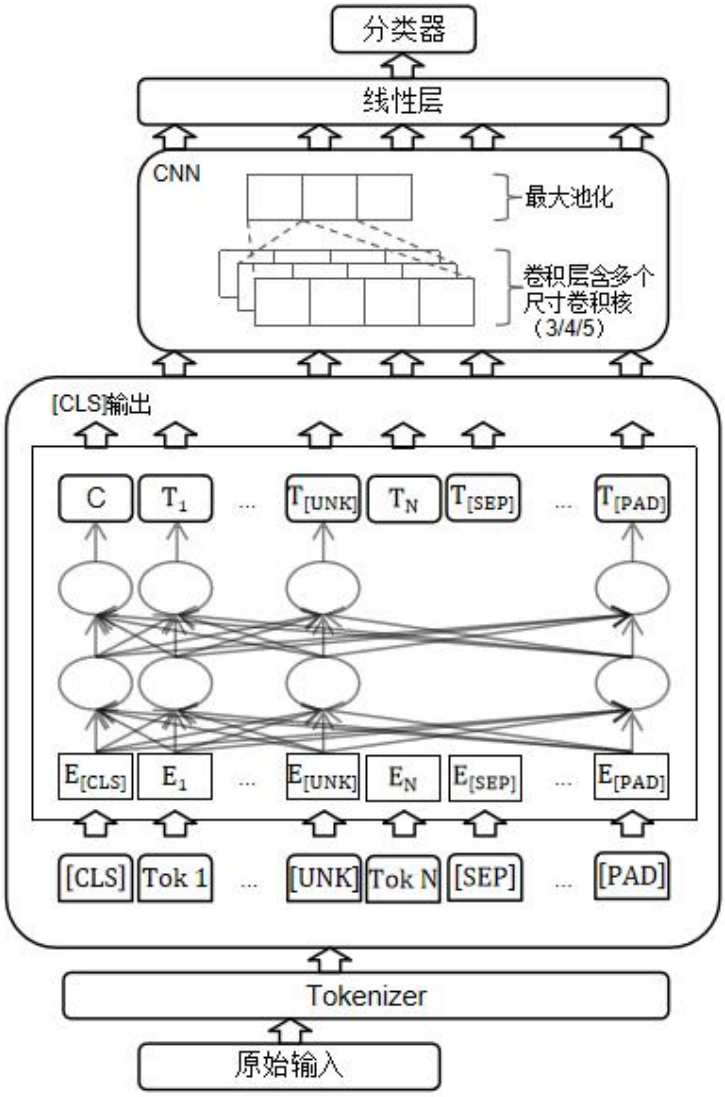

图 2-12 添加 CNN+线性层结构

（3）最后添加 BiLSTM 层和线性层，如图 2-13 所示，和 CNN 同样类似，但三个 CNN 层被一个双向的 LSTM 层所取代[54]，BiLSTM 是传统语言表示方法中的代表，能够捕获序列中的位置信息，这恰恰是预训练语言模型所抛弃的。简而言之，语言模型的每一层的输出被输入入到 BiLSTM 神经网络，然后将输出的最后两个隐藏状态连接起来，再送入最后的线性分类层。

另外，添加的层的参数是随机初始化的，在微调的训练过程中更新权重。CNN 和 BiLSTM 是在预训练语言模型时代之前的文本表示方法，其中 CNN 是基于感受野捕获固定窗口内的词的语义表示；而 BiLSTM 是利用双向上下文循环的结构，表示上下文语义信息。

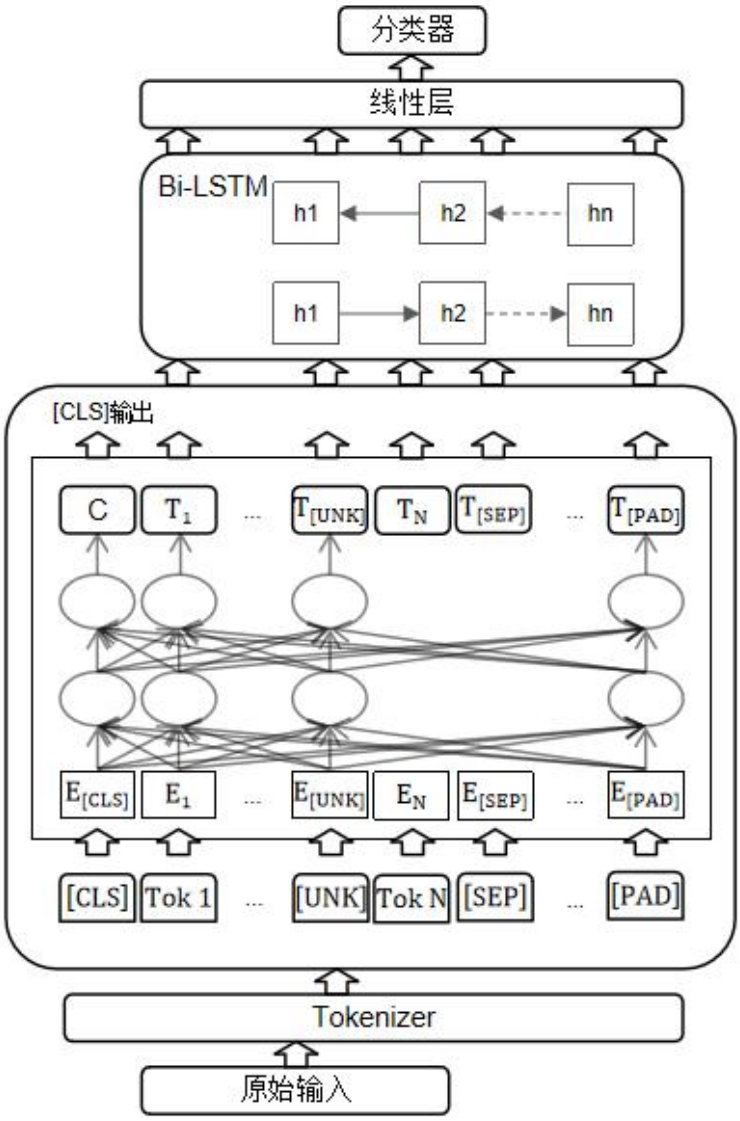

图 2-13 添加 BiLSTM+线性层结构

2.3 微调训练细节

为了对比不同模型的表现，对比同类模型不同结构的表现。遵循消融实验的原则，采取原论文推荐的参数设置微调，而相同模型添加不同结构采取与基础模型相同的

参数设置训练，添加的同种下游网络结构参数一致。

基础 BERT 模型采用相同的参数，对应文件 config.json，数据预处理采用相同的 BERT 的 Tokenizer 配置，对应文件 tokenizer_config.json，其中还需要导入 Tokenizer 使用的词表文件 vocab.txt，Token 索引映射文件 tokenizer.json。模型采用预训练的权重参数初始化，对应文件 pytorch_model.bin。另外研究重点尚未关注其它语言的数据对当前下游 TCC 任务的影响，尽管可能由于其它语言的 Token 被映射为 OOV 词，不失一般性，采取通用的 bert-base-case 版本模型。具体参数如表 2-1 所示。

表 2-1 BERT 模型配置

模型配置	参数
architectures	[“BertForMaskedLM”]
attention_probs_dropout_prob	0.1
gradient_checkpointing	false
hidden_act	“gelu”
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
layer_norm_eps	1e-12
max_position_embeddings	512
model_type	“bert”
num_attention_heads	12
pad_token_id	0
position_embedding_type	“absolute”
transformers_version	“4.6.0.dev0”
type_vocab_size	2
use_cache	true
vocab_size	28996

基础 RoBERTa 模型采用相同的参数，对应文件 config.json，数据预处理采用相同的 RoBERTa 对应的 Tokenizer。预训练结果中的存储 BPE 过程中得到的所有 Token 对应文件 merges.txt，可以简单理解成就是字典。而词表到索引的映射对应文件

vocab.json，根据 merges.txt 将输入的文本 Token 化，再根据 vocab.json 中的字典映射为对应的索引。模型采用预训练的权重参数初始化，对应文件 pytorch_model.bin。

前面提到 RoBERTa 丢弃了 NSP 任务，Tokenizer 处理过程略微不同，词表大了许多，这对某些词的 OOV 问题有一定改善作用。具体参数如表 2-2 所示。

表 2-2 RoBERTa 模型配置

模型配置	参数
architectures	[“RobertaForMaskedLM”]
attention_probs_dropout_prob	0.1
bos_token_id	0
eos_token_id	2
hidden_act	“gelu”
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
layer_norm_eps	1e-05
max_position_embeddings	514
model_type	“roberta"
num_attention_heads	12
num_hidden_layers	12
pad_token_id	1
type_vocab_size	1
use_cache	true
vocab_size	50265

基础 XLM 模型采用相同的参数，数据预处理采用相同的 XLM 对应的 Tokenizer，对应文件 tokenizer_config.json，merges.txt 文件、vocab.json 文件和 RoBERTa 类似。模型采用预训练的权重参数初始化，对应文件 pytorch_model.bin。XLM 的 Tokenizer 和 BERT、RoBERTa 采用的单语种版本不同，同样对于目前使用的微调语料库，主要是英文、及拉丁语系，其中的社交平台里大部分其它语种为法语。具体模型参数如表 2-3 所示。

表 2-3 XLM 模型配置

模型配置	参数
architectures	[“XLMWithLMHeadModel”]
asm	false
attention_dropout	0.1
bos_index	0
bos_token_id	0
causal	false
dropout	0.1
emb_dim	1024
embed_init_std	0.02209708691207961
end_n_top	5
eos_index	1
gelu_activation	true
pad_token_id	0
id2lang	{"0": "en", "1": "fr"}
init_std	0.02
is_encoder	true
lang2id	{"en": 0, "fr": 1}
layer_norm_eps	1e-12
mask_index	5
mask_token_id	0
max_position_embeddings	512
max_vocab	-1
min_count	0
model_type	"xlm"
n_heads	8
n_langs	2
n_layers	6
pad_index	2
pad_token_id	2
same_enc_dec	true
share_inout_emb	true
sinusoidal_embeddings	false

续表 2-3

start_n_top	5
summary_activation	null
summary_first_dropout	0.1
summary_proj_to_labels	true
summary_type	first
summary_use_proj	true
unk_index	3
use_lang_emb	true
vocab_size	64139

XLM 具体 Tokenizer 配置如表 2-4 所示，注意到 lang2id 参数即前面提到的，语言嵌入，用于判断当前位置的 Token 是什么语种，这里使用的模型版本只有英语和法语，在模型最后一层输出时，再根据 lang2id 参数映射成对应语种的词。

表 2-4 XLM 的 Tokenizer 配置

Tokenizer 配置	**参数**
model_max_length	512
do_lowercase_and_remove_accent	true
id2lang	{"0": "en", "1": "fr"}
lang2id	{"en": 0, "fr": 1}

另外，由于是在下游 TCC 任务上微调训练，而基础语言模型的参数是使用预训练的参数初始化的，即上面提到的权重文件件 pytorch_model.bin。而添加下游网络结构：线性层、CNN+线性层、BiLSTM+线性层的时候，这些层的权重是随机初始化的，因此为了充分训练以发挥它们的性能，调试后采取最佳 30 个轮次的微调训练。

模型使用的分词方法往往不是直接的词级别或字符级别的分词。前者会使得词表过大，后者则表示能力太低。因此主流的方式是进行子词级别的分词，例如对于“tokenizeation”这个词，可能会被分成“token”和“ization”两部分子词。常见的子词 tokenization 方法有：BPE、WordPiece、Unigram、SentencePiece。子词分词的核心思想就是：出现比较频繁的词就不应该被切分成更小的单位，但不是经常出现

的词应该被切分成更小的单位。这样分词的好处是，大大节省了词表大小，而且还能解决一定程度的 OOV 问题，例如某些未知词拆成子词后能够在词表里找到，就不会被映射成[UNK]Token。因为很多使用的词汇，都是由更简单的单词或前缀 、后缀构成的，不去保存这些小的子词的各种排列组合成的千变万化大词，而用较少的词汇去覆盖各种情况的词表示。同时，与直接使用最基础的字符作为词表相比，子词的语义表示粒度更完整，表达能力也越强。

在数据处理过程，由于按最大文本长度去做 Tokenizer 操作，再输入到模型。传统的处理方式是将所有的 Token 对齐成最大长度。这里提出一个高效处理数据的的技巧，使用 map 方法将实例化后的 tokenizer 方法映射到每条数据。通过这个 map，被处理的数据会被缓存，所以即使重新执行代码，也不会耗费时间。同时能够使用多进程来处理从而提高处理的速度。不需要把整个数据集全部加载到内存里，同时每条数据一经过处理，就会马上被保存。

使用动态 padding 的策略，在 Tokenizer 操作中，故意先不进行 padding，因为想要在划分批次的时候再进行 padding 操作，这样可以避免出现很多有一堆 padding 的序列，每个批次按照这个批次的最大句子长度 padding，不同批次的 padding 长度不同，这样可以最大限度节省显存。使得可以在调参的过程中最大限度利用显存，而由第 2.3 节的 RoBERTa 的结论，大的批次大小，有利于提升模型训练的表现，其原因是更新梯度的过程中使用小批次梯度下降，用一个批次的数据分布近似整体数据分布的损失，计算梯度更新，而批次更大，代表整体分布也更公平，近似也更准确。

训练循环中的评估策略采取每 10 个步长评估一次，这个评估频率比较合适，不会太费时间。

与原始模型训练使用的 Adam 学习率优化算法相比，为了缩短训练时间，采用后续出现更快收敛的 AdamW 优化算法。

设置梯度裁剪因子为 1.0，用来控制梯度的最大值，避免梯度过大，使得权重变化过大，从而让模型不稳定。

两张显卡采用分布式数据并行的方式训练，可以多进程工作在多个显卡中。相比于过去经常使用单进程将模型和数据加载到多个显卡上，而损失却在一个显卡上

计算导致负载不均衡的数据并行多卡训练的方式而言，分布式数据并行为每个显卡都创建一个进程，既可用于单机多卡，也可以多机多卡训练，每个进程执行相同的任务，每个进程都和其它进程进行通信。另外一点不同的是，只有梯度才会在进程之间传播。以单机多卡为例，假如三张显卡并行训练，那么在每个轮次中，数据会被划分成三份，分给三个显卡，每个显卡使用自己的小批量的数据做各种的前向传播计算，然后梯度在显卡之间通过进程间通信共享梯度，整合梯度，然后在各种独立的更新参数。

2.4 本章小结

本章分析了TCC任务，使用的整体模型结构和三种对下游网络改进结构，在TCC任务上直接进行微调，添加下游网络结构改进后再微调，以及微调的训练策略。

3 TCC 任务领域内继续预训练策略

本章主要介绍了从训练数据领域的角度去优化模型，首先讨论了不同领域的数据分布对模型效果的影响，然后从领域数据和目标任务数据入手提出在下游 TCC 任务上去优化模型效果的策略。

3.1 数据分布对模型效果的影响

对来自各种来源的文本进行预训练的语言模型构成了当今 NLP 的基础。鉴于这些广泛覆盖模型的成功，Gururangan[25]等人研究了在目标任务领域内定制预训练模型是否仍然有帮助，提出了一项涉及四个领域（生物医学和计算机科学出版物、新闻和评论）和八个分类任务的研究，表明在领域内继续预训练的第二阶段（领域自适应预训练）会提高在高资源和低资源环境下的收益。此外，即使在领域自适应预训练之后，适应任务的未标记数据（任务自适应预训练）也能提高性能。使用简单的数据选择策略扩大的任务语料库是一种有效的替代方案，特别是当领域自适应预训练的资源可能不可用的时候。总体而言，多阶段自适应预训练在任务效果方面提供了很大的收益。

预训练语言模型是在大规模、不同种类的语料库上进行训练的[58, 59]。例如，RoBERTa[22]使用了超过 160 GB 的未压缩文本来训练，来源从英语百科全书和新闻文章到文学作品和网络内容。通过这样的模型学习的表示在许多任务中实现了强大的性能，使用来自各种来源的不同大小的数据集[44, 60]。由此提出问题：一个任务的文本数据领域，通常用来表示某一特定文本主题或风格（如科学或小说），语料库的数据分布相关性如何？最新的大型预训练的模型能否普遍适用，或者为特定领域构建单独的预训练的模型是否仍然有帮助？对于本文中的四个 TCC 任务的数据集，语料分布集中在评论领域，与预训练模型所用的原始语料库的分布存在一定的相似度。

虽然一些研究显示了对特定领域的未标记数据进行持续预训练的好处，例如 Lee 等人[26]，但这些研究一次只考虑一个领域，并使用一种在较小且不太多样化的语料库上进行预训练的语言模型，而不是最近的语言模型。此外，也不清楚继续预训练

的好处如何定量描述，比如随可用标签任务数据量或目标域与原始预训练语料库的接近之类的因素而变化，如图 3-1 所示。

其中以 RoBERTa 为例，考虑四个领域（生物医学和计算机科学出版物、新闻和评论），以及八个分类任务（每个领域两个）。对于尚未在 RoBERTa 语料领域中的目标任务，在相关的领域上进行持续的预训练，即领域自适应预训练 DAPT（Domain-Adaptive Pretraining），在高资源和低资源环境下，都能持续提高目标领域的任务性能。这为在 TCC 任务上应用带来启发，如果能够获得大量的目标领域的无标注语料，这往往仅需要简单的数据抽样处理，成本非常低，便有可能获得任务效果的提升。类比到本文所用的 BERT、XLM 模型，尽管预训练的语料库存在差异，结论仍值得参考。

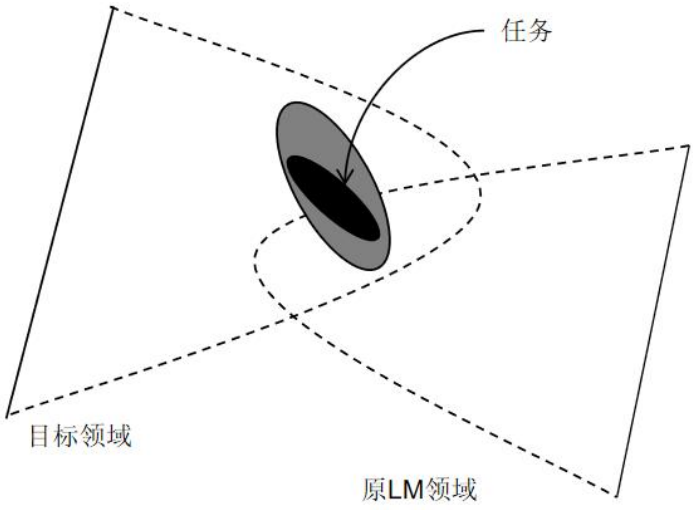

图 3-1 数据分布的图解

任务数据由可观察的任务分布组成，通常从更大的目标领域内的更宽的分布(黑色虚线内)中非随机采样，该目标领域不一定是原 LM 预训练领域中包括的领域之一即使可能存在重叠。在任务分布和领域分布中的数据上，继续预训练是否也能够重复利用这些数据分布的表示？主要考虑围绕群体和论坛的语料领域，不过也可以从用于任务的给定语料库中归纳出领域，例如在模型的有监督训练中使用的领域。这就提出了一个问题，即在与任务更直接相关的语料库上进行预训练，是否可以进一步提高模型的效果。Gururangan[25]等人在一个较小但是与任务语料直接相关的语料库

上研究了领域自适应预训练与任务自适应预训练 TAPT（Task-Adaptive Pretraining）的比较，使用来自任务分布的未标记任务数据集。任务自适应预训练已经被证明是有效的[37]，但通常并不用于最新的模型。无论有没有领域自适应预训练，TAPT 都能大大提高 RoBERTa 的性能。当从任务设计者或注释者手动管理的任务分布中获得额外的未标记数据时，任务自适应预训练的好处会增加。

自 2018 年以来，大多数 NLP 研究系统的学习包括两个阶段的训练。首先，在大型的未标记语料库上训练语言模型 LM，该模型通常具有数百万个参数。在预先训练的模型中学习的词或子词片段[38]表示，而后在下游任务的监督训练中重复使用，并从第一阶段对表示和网络进行可选的微调。其中之一就是 RoBERTa，它使用与其前身 BERT 相同的基于 Transformer 的结构，以及掩盖语言模型的任务目标（即预测随机掩盖标记时的交叉熵损失）进行训练。RoBERTa 的未标注预训练语料库包含来自不同英语语料库的超过 160 GB 的未压缩原始文本。RoBERTa 在各种各样的任务上取得了比它的前面的工作有更好的表现，使其成为选择的基线。尽管 RoBERTa 的预训语料库来自多个来源，但这些来源是否足够多样化，是否足以概括英语中的大多数变化还没有确定。RoBERTa 的训练语料领域之外的是什么。为此在两类未标注的数据上对这个大语言模型进行继续预训练来一步适应：（1）特定领域文本的大型语料库；（2）以及与给定任务相关联的可用的未标记数据。

3.2 领域自适应预训练

领域自适应预训练 DAPT 方法很简单，继续在大量未标记的特定领域文本语料库上对 RoBERTa 进行预训练。对四个被使用最多的数据领域：生物医学(BioMed)论文、计算机科学(CS)论文、RealNews 的新闻文本和亚马逊商品评论文本。之所以选择这些领域，是因为它们在以前的工作中很受欢迎，并且每个领域都提供了用于文本分类的数据集。

为了分析领域相似度，在执行 DAPT 之前，首先需要量化目标领域与 RoBERTa 的预训练领域的相似度。

考虑在每个领域的语料库中挑选出的文档中抽样大小相当随机样本中包含前 1

万个最频繁的一元语法 Unigram（即以单个连续词为一个单元，N 个连续单词为 N 元语法），组成不包括停用词的领域词汇表。这些样本的大小相当。除了评论之外，对每个领域使用 5 万个挑选出的文档，在评论中使用 15 万个挑选出的文档，因为它们要短得多。

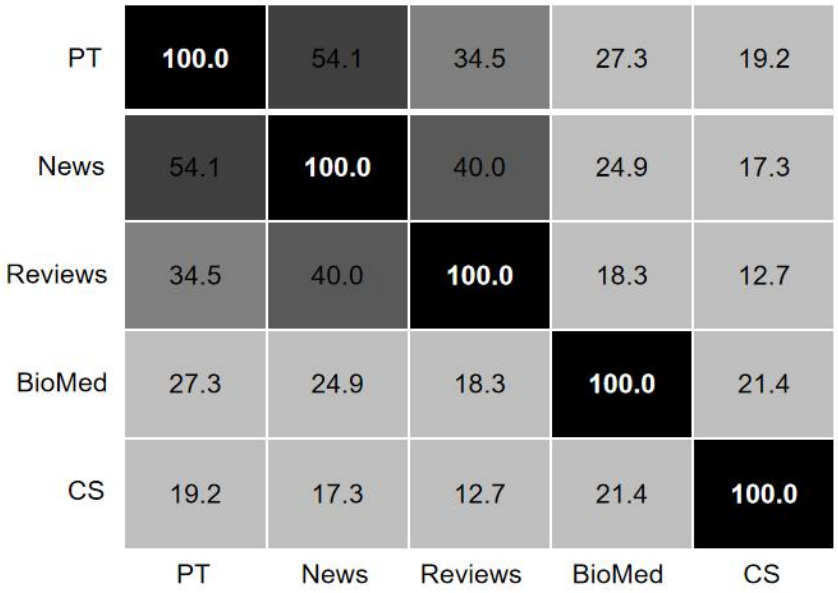

图 3-2 领域之间的词汇重叠比例(%)

另外还从与 RoBERTa 的预训练语料库（即 BOOKCORPUS、 STORIES、维基百科和 REALNEWS）类似的来源中抽取了 5 万个文档来构建预训练领域词表，因为原始的预训练语料库没有发布。图 3-2 显示了这些示例领域之间的词汇表重叠，PT 表示样本来自与 RoBERTa 的预训语料库相似的来源，每个领域的词汇表是通过考虑从每个领域采样的文档中最频繁的 1 万个单词（不包括停用词）来创建的。可以观察到，RoBERTa 的预训练领域与新闻和评论有很强的词汇重叠，而 CS 和 BioMed 与其他领域的差异要大得多。这一简单的分析表明，RoBERTa 适应不同领域预训练的预期收益程度，领域越不相似，通过 DAPT 提升效果的可能性就越高。

要考虑任务数据如何分配到特定领域，对 DAPT 的分析就必须考虑到领域重叠。例如，为了在获得帮助评论上执行 DAPT，只在亚马逊的评论做自适应预训练，而不是任何评论文章。然而，如图 3-2 中的渐变表明，领域之间的边界在某种意义上是模糊的。例如，40%的 Unigram 在评论和新闻之间共享。作为这种领域重叠的进一步

推论，定性地识别跨域重叠的文档。事实上，使 RoBERTa 适应新闻领域，并不会对其在评论任务中的表现造成损害。

虽然这一分析并不全面，但它表明，引起可观察到的领域差异的因素可能不是相互排斥的。超越传统领域边界的预训练可能导致更有效的 DAPT。一般而言，在设计训练前程序和创建测试领域外推广能力的新基准时，必须牢记数据的来源，包括对语料库进行管理的过程。

3.3 任务自适应预训练

为了获得感兴趣的特定任务而专门制作的数据集，往往只会覆盖更广泛领域内文本的子集。假设，在任务数据是更广泛领域的狭义子集的情况下，对任务数据集本身或与任务相关的数据进行预训练可能是有帮助的。

任务自适应预训练 TAPT 是指在给定任务的未标记训练集上进行预训练；以前的工作已经证明了它的有效性[37]。与领域自适应预训练相比，任务自适应预训练达到了一种不同的权衡：它使用的预训练语料库要小得多，但是与任务相关性更高（假设训练集很好地代表了任务的各个方面）。这使得 TAPT 的运行成本比 DAPT 低得多，TAPT 经常能与 DAPT 性能相当。

TAPT 持续改进 RoBERTa 在跨领域所有任务的基线，即使在 RoBERTa 预训练语料库中的新闻领域，TAPT 也比 RoBERTa 有所提高，显示出任务自适应预训练的优势。特别值得注意的是 TAPT 和 DAPT 之间的相对差异。DAPT 的资源密集度更高，但 TAPT 在某些任务中的性能与其相当。

即使是一个由数亿个参数组成的模型也难以编码单个文本领域的复杂性，更不用说所有语言了。针对特定任务或小型语料库对模型进行预训练可以提供显著的好处，在不断扩大的语言模型上进行各种任务的同时，努力识别和使用与领域和任务相关的语料库来针对性继续预训练模型是有价值的。

3.4 轻量 TAPT

在这一部分中，本节探索了在 TCC 任务上使用最小计算资源进行领域内继续预

训练的性能。在 TAPT 工作的基础上进行有限资源下的 TAPT，并减少 TAPT 的超参数设置以进一步减少所需的计算资源，将其称之为轻量 TAPT[25]。更具体地说，批量大小减少到 16，而原来的 TAPT 是 2048。其次，相比与原始 TAPT 中的 100，分别尝试训练不同数量的轮次：1，5， 10， 20， 50，100。

总之，对于每个轻量 TAPT 模型，训练过程分为两个阶段。

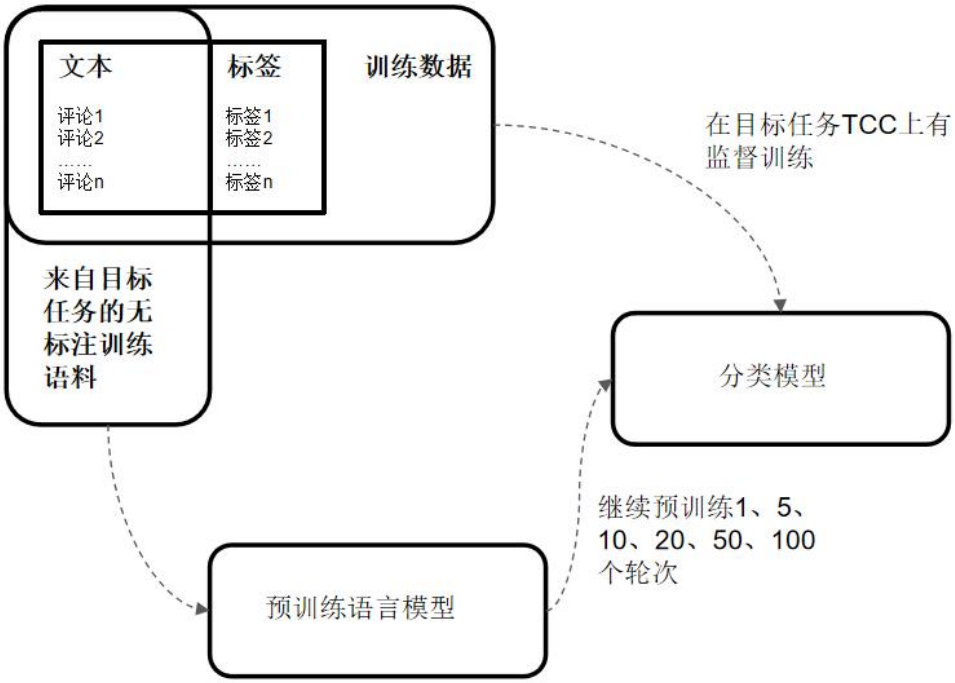

图 3-3 轻量 TAPT 模型训练过程

在第一阶段，如图 3-3 中的所示，预训练的语言模型在无标注的训练语料库上进行继续预训练，该语料库来源于下游 TCC 任务。分别对进行模型训练不同的轮次，以便后面获取训练轮次的影响。

在第二阶段，如图 3-3 中的所示，语言模型的输出层被替换为线性分类层，这应用第 2 章得出的最佳下游结构，然后在 TCC 任务上以有监督的方式微调训练新的分类模型，所有的参数都是共同训练的。只有线性分类层的参数是随机初始化再从头开始训练的。

注意一般预训练要求的语料库的规模以及所需的计算资源都非常大，往往不是个人研究者所能达到的条件，因此考虑在简化硬件资源以及数据的规模，使得利用

任务自适应预训练带来下游任务效果的收益。

同时为了实验的一致性，对不同数据集采取了相同的配置。另外为了便于在当前有限的实验环境下进行预训练，采取了和第 2.5.3 小节中微调训练一致的策略，细节上与原论文略有不同。使用 AdamW 优化器、梯度累计等训练加速收敛优化等技巧，和前面微调的训练相同。

注意到第 2 章中各预训练模型的预训练任务，BERT、RoBERTa 和原来的训练任务保持一致。

其中 BERT 仍然采取原来的 MLM 和 NSP 预训练任务，对语料数据的处理以句末标点符号分句，连续的两句作为一个句对的输入。随机遮盖的比例和原来相同，随机遮盖与替换操作提前执行一次，并复制多份来使得同一个句子能够有不同的遮盖，并在训练阶段保存，训练中喂给不同的轮次。使用的词表还是字符级别的 BPE 子词词典。

而 RoBERTa 去掉了 NSP 只有 MLM 预训练任务，对语料数据处理按照最大长度截断成一句作为输入，不过不跨文档。调整在 MLM 任务中的静态掩码机制，采取更高效训练的动态掩码机制，具体来说每一次将训练语料喂给模型的时候，再进行随机遮盖的操作，每次都会生成新的遮盖模式，对数据利用的效率更高，同一个句子在不同轮次中的遮盖部分更加多样，学习不同的语言表示。使用是词表是字符级和词级别表征的混合的 BPE 子词词典。

XLM 略微不同，在原论文的预训练中对各语种进行高低资源的多项式分布采样，这需要已知数据集的语种分布，而本文使用的公开 TCC 任务数据集并未给出这些语种分布的具体信息，仅有大致包含哪些语种，因此根据语种分布采样无法做到。另外在所用的毒性评论任务数据上进行预训练，对于 TLM 任务而言，依赖有监督的两种语种的语义对齐句子对作为训练数据，而该数据集并不能提供，暂不考虑，由此带来了略微的不一致。

3.5 本章小结

本章介绍了数据分布对模型效果的影响，分析领域内和目标任务对模型继续预训练的条件，最后提出在下游 TCC 任务上使用较低计算资源进行轻量 TAPT 来提升模型效果的策略。

4 模型压缩部署策略

本章主要介绍目前大规模预训练模型的部署问题，首先简介了一下近年来各种超大参数规模的模型，然后分析了由此带来落地的问题，介绍了几种模型压缩的基本方法，并分析指出了应用场景，最后选择了最合适的蒸馏策略用于 TCC 任务，提出一种浅而宽的 3 层 Transformer Block 和 1024 维隐层维度的结构的学生模型，多阶段蒸馏的策略。

自从 2018 年，Google 提出的 BERT 模型，参数量达到了 340M，即有 340M 个神经元，导致了 CPU 根本加载不起来；而到了 2020 年，Open-AI 发布的 GPT-3[61] 模型，参数量达到了惊人的 178B，连内存都无法加载进来，无法部署在现有服务器中，没有办法进行推断。为了将模型部署到手机、智能硬件等资源有限等设备上，动辄上亿参数量的大模型，显然没办法。预训练语言模型参数量进入暴涨期，语言模型的预测与部署成为了核心问题。模型规模的大规模增长，性能也随之提高，这导致了较高的推理时间和存储成本。这成为工业应用的主要障碍，特别是在边缘设备上部署。

模型压缩基本方法分为三类：（1）量化，需要与硬件实现紧密结合；（2）裁剪，大多数时候只能实现压缩，而无法提升预测速度；（3）蒸馏，需要重新进行训练。

4.1 浮点数权重量化

量化就是将一个模型的浮点数换成整数，一个浮点数占 4 个字节，而一个 int8 整数占 1 个字节，将权重的浮点数都换成整数就能省 4 倍显存。但是浮点数是连续的，而 int8 整数是离散的，直接替换的话肯定会带来精度损失，需要在浮点数里找到一个连续的数，比如某一片数域的中心点和一个整数对应上，预测时再还原回来。神经网络中的基本运算就是在进行矩阵运算，量化将浮点数量化成一个浮点数系数乘一个整数，而矩阵中的系数是共享的，将浮点数的乘法变成整数的乘法。整数的矩阵乘法在很多硬件，主要是手机端的硬件比如 ARM 架构上，都有高效的实现，很

多时候都比浮点数乘法要快，但是在 GPU 上，有些时候是反过来的，所以说量化要看硬件，但整体来说，参数量是压缩了 4 倍。只有线性量化才能提升推断效果，否则只能退化成查表。在但是量化后，模型的精度受到了很大的损失，量化后的模型一般需要 retrain 再训练的操作，来保持精度，将精度校准回来。因为网络已经量化成了一个整型，要对一个整型的网络求导，直接做是做不了的。其反向过程只能采用直通估计法处理。

而对于目前的 TCC 任务，在各大互联网内容平台，往往是封装成服务部署在 GPU 或 CPU 集群上。由于硬件架构缺乏专门的算子优化，表现也不稳定，一般也比较麻烦，不太常用。

4.2 冗余结构裁剪

在神经网络中，基于这样的观察，在神经网络有大部分权重是接近 0 的值，一些 0 点几数字的权重，完全就能将他们删掉，其实对网络对预测是没有什么影响对 。在计算机视觉 CV 领域有较为显著的应用。但是假如说对全部权重进行一遍统计，单纯地对小于某个阈值的数值砍掉，权重矩阵会产生不规则的空洞。也就是说，随机裁剪会形成稀疏的权重矩阵，从而不利于预测提速，所以得进行结构化的裁剪。按整行或者按整列裁剪，那样，权重矩阵的维度就可以降低了，矩阵乘法运算也会高效一点。然而结构化的裁剪又通常会导致精度下降。结合现在的场景，做预训练模型的裁剪，如何做结构化的裁剪？回顾 Transformer 的结构，如图 2-4 所示，可以发现，Transformer 结构的多头自注意力 Multi Head Attention 机制，这个操作本身就是一个易于裁剪的结构。回顾一下直接介绍 Transformer 结构，会发现计算 Attention 矩阵的时候，缩放点积随机定型的堆叠了好几块。如果将其整个块删除那么对应的就是前面说的结构化裁剪，从而得到一个结构化的矩阵，对训练会有更加高效的实现。因此，就需要对 Head 裁剪，有相关工作表明，以 BERT 为例有 12 个 Head，随机裁剪某些 Head，模型在任务中的准确率并没有下降多少；而有些关键的 Head 被裁剪掉后，模型在任务中的表现却会下降很多；裁剪掉某些 Head，模型在任务中的表现反而会提升。整体还是说明，模型的 Head 其实是冗余的,因此可以对 Transformer

的 Head 进行裁剪。那么，怎么进行裁剪呢？其实不是随机的进行裁剪。一个比较好的方案是，先对 Head 的重要性进行排序，按重要性从小到大裁剪。Head 的重要性定义为：去掉一个 Head 之后，模型损失函数的变化。如公式（4.1）所示，I_i表示第 i 个 Head 的重要性，c_i为第 i 个 Head 的系数。

$$I_i = \frac{\partial L}{\partial c_i} \tag{4.1}$$

同样地，虽然裁剪可以用于预训练语言模型，不过需要事先对各个 Head 进行多轮裁剪计算重要性排序，才能找到冗余的结构。然而往往像 BERT 这样的大模型有 8 个 Head，至少每层得进行 8 轮裁剪实验，才能得到各个 Head 的重要性排序，非常麻烦，实际中也很少应用，因此在下游 TCC 任务上也很少使用。

4.3 多阶段模型蒸馏策略

本节首先介绍了知识蒸馏，分析得出适用于当前的 TCC 任务，然后提出了多阶段模型蒸馏过程，一个浅而宽的学生模型，以及训练的具体步骤。

4.3.1 知识蒸馏

知识蒸馏是目前比较主流的预训练模型的压缩方案。在蒸馏的过程中需要引入

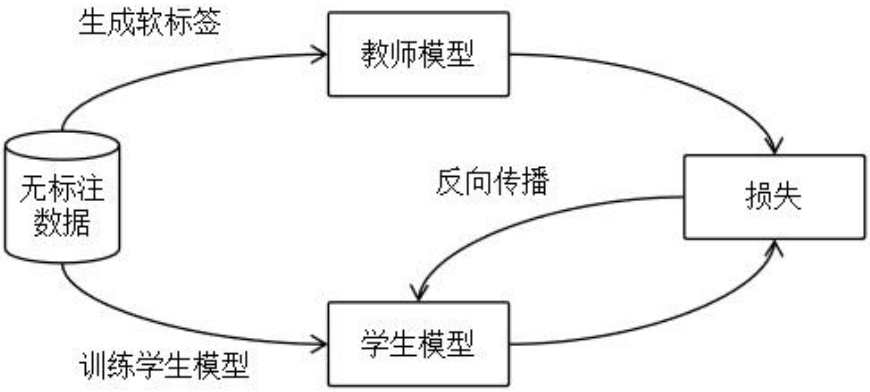

图 4-1 模型蒸馏过程

一个教师和学生的概念，其实是用教师模型去调教学生模型。具体来说，如图 4-1 所示，模型蒸馏的过程中需要两个模型，一个叫做教师模型，一个叫做学生模型。教师模型是一个已经训练好的模型，也就是说已经在下游任务中获得了不错的效果

的模型。学生模型是一个重新初始化的小模型，去学习教师模型的结果，因为不是真实的标签，而是教师模型预测的结果，也称为软标签。

一般来说，学生模型可以是词袋模型、CNN、LSTM 等小型模型，而教师模型往往采取 BERT 类大模型。蒸馏的过程为，将一个数据同时输入给学生模型和教师模型，会各自得到一个概率分布，有学生模型输出的概率分布和教师模型的概率分布。假设目标是让学生模型的概率分布尽可能去拟合教师模型的概率分布。通过设置损失函数让学生模型的分布向教师模型上靠，使用这种方式使得学生模型能够复现教师模型的输出，从而也能获得教师模型的准确率，这是蒸馏的训练方式。具体来说，训练的损失函数就是学生模型输出的概率分布和教师模型输出的概率分布的距离，一般用 KL 散度来刻画他们的距离，损失函数就是最小化这个距离。除此之外，假如数据有真实的标注的话，还可以学习真实数据标注的分布，同时进行两个学习任务。

知识蒸馏主要改进方向有两个。蒸馏的信号：除了教师模型的输出的概率分布，还可以学习中间层的表达；蒸馏的流程：如何在当今的预训练加微调的范式中去做蒸馏。

其中就预训练语言模型而言，蒸馏的信号可以选择一些更多的中间结果。例如对于 Transformer 结构而言，教师模型可以选择大的 BERT 类预训练模型，学生模型可以选择层数少一点的，比如 3 到 4 层 Transformer 结构的小模型。由于教师模型和学生模型是同种结构，中间结果也能用于作为蒸馏信号。自顶向下：（1）如果数据集有标注，即硬标签。（2）和上面所说的一致，使用教师模型输出的概率分布为软标签。这两个一般是用交叉熵损失来度量，损失函数分别为学生模型输出概率分布和真实标签的交叉熵距离，以及学生模型输出概率分布和教师模型输出概率分布即软标签的交叉熵距离。（3）当学生模型也是和教师模型的基本结构一样的多层 Transformer 结构的时候，每层都有个隐藏状态的输出的中间信号，也可以鼓励学生模型向教师模型靠近，但是隐藏状态一般不是一个概率分布，一般是一个高位空间的向量，而对向量之间的距离进行刻画，一般采用的是均方差损失，也就是他们的欧式距离。（4）Transformer 中的 Self-Attention 结构，其实刻画的是输入两两词之

间的相似度，就是一个重要性矩阵，也能作为指引信号来用于蒸馏，采用均方差距离刻画。

4.3.2 多阶段蒸馏

就蒸馏的流程而言，如何将蒸馏的流程与现在的预训练加微调的范式结合起来做考虑。那么是预训练蒸馏，还是微调蒸馏，还是都蒸馏？如图 4-2 所示，一个最全面的流程是：学生模型先进行自己的学习，也就是 A 这一步，一般是无监督数据上进行 MLM 这样的自监督任务训练。训练好了之后，再到 B 这一步，加入一个教师模型，一般是性能更强的预训练模型，同样也是在无监督的语料上进行蒸馏，通过

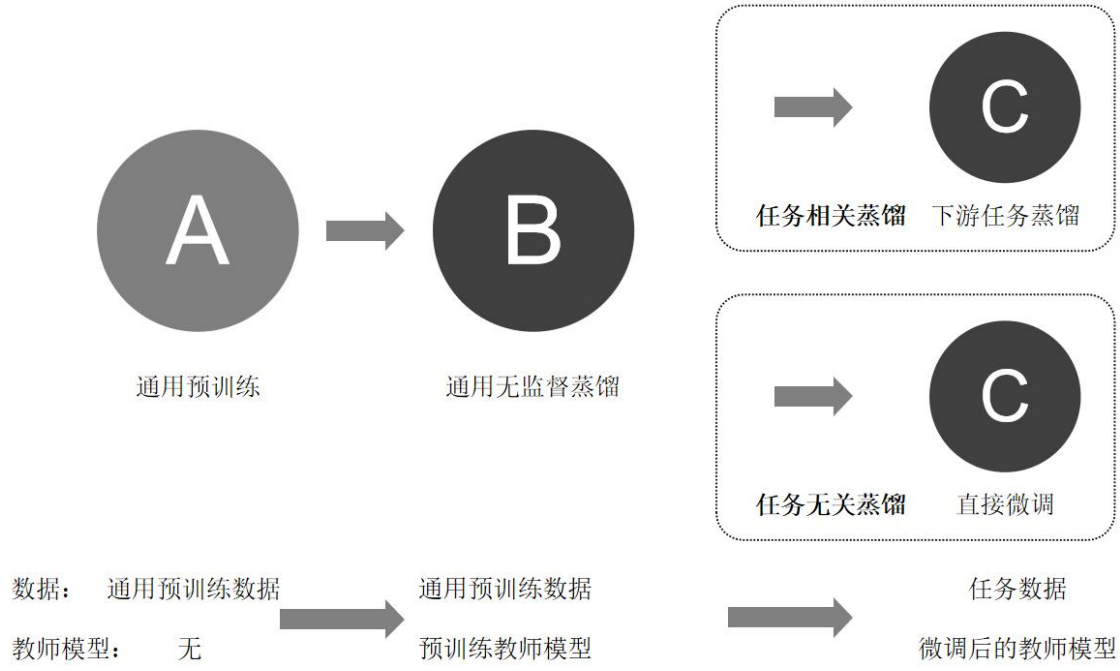

图 4-2 蒸馏的流程

通用的数据上对学生模型进行迁移，学习完就能够拿去微调。这种蒸馏称为任务无关蒸馏，不需要知道下游任务数据，好在方便，一般效果不错，但是效果会差一点。另一种，为了获得更好的效果，需要任务相关蒸馏。也就是需要下游任务数据，利用下游任务数据微调教师模型，对经过了两轮蒸馏的学生模型继续进行蒸馏，这样一般学生模型就能达到最好的准确率。

这和第 3 章中的领域内继续预训练的思想是一致的，蒸馏是用特殊信号的训练，在逐步向任务领域靠近后，性能也能提升。一般而言，模型蒸馏就是以一定性能的

损失为代价，换取模型规模的轻量化，推理速度的提升，以及存储的大小下降。采用继续在目标任务领域上蒸馏，可以在学生模型规模确定的情况下，恢复一定的性能。

因此，本文提出了一种多阶段递进蒸馏方法来压缩预训练语言模型，该框架通过三个组成部分将学生模型从通用级别逐步微调到特定任务级别。具体地说，第一个阶段是一般蒸馏，在预先训练的教师模型、通用的数据集和潜在蒸馏损失的指导下进行蒸馏。然后，进行普通增强蒸馏将教师模型从预训练教师转变为微调后的教师模型。之后，任务自适应蒸馏算法将训练数据从通用数据集转换到特定下游任务数据集，增加了两个额外的损失，即软标签损失和硬标签损失。

对于获取的性能最好的模型，采取类似领域继续预训练和微调的思路进行任务相关蒸馏，在实验的设置中，除了有标注的数据集外，还使用了大规模的无标注数据，称之为通用数据集，教师模型可以在这些数据集中进行预训练和微调。即前面的工作。这些未标记的数据可以很容易地从在线爬虫网站获取，如维基百科，并使用在预训练的语言模型中。

4.3.3 学生模型设计及训练

为了将这些数据和教师模型的知识顺利地结合起来，本文设计了一个多阶段的蒸馏步骤，依次包括通用数据预训练、通用数据蒸馏、任务相关蒸馏，共三个阶段持续蒸馏。

如图 4-3 所示，模型采取了 3 层的 Transformer 结构，蒸馏的指导信号，采用输出的预测分布软标签，以及中间层 Transformer Block 的隐藏状态输出，与原版 BERT 相比，使用了更浅的层数，从 12 层减少到了 3 层 Transformer Block，理论上线性提速 4 倍；而每一层还采用了更宽的 Transformer Block，更大的 Hidden Size，从 768 维增加到 1024 维，来弥补模型变浅带来的效果损失。通过知识蒸馏技术，进一步提升模型的预测速度。

蒸馏训练的过程和前面第 3 章类似，首先在将学生模型在和教师模型预训练使用的通用语料库一致的数据训练预训练任务；然后还是在通用语料库上，使用教师模型的 1、6、12 层的 Transformer Block 的输出，以及预测结果软标签作为指导信号，

蒸馏训练学生模型，轮数与预训练一致；最后在下游 TCC 任务上，额外增加一个真实标注加入指导信号，蒸馏学生模型，得到最终的预训练模型的蒸馏 Tiny 版本。

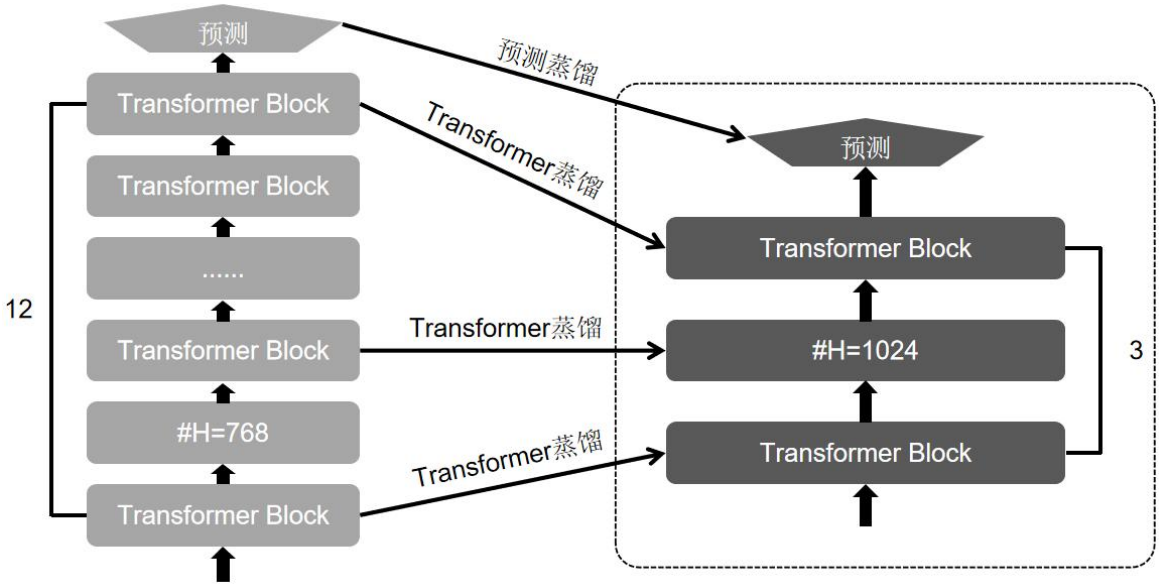

图 4-3 模型蒸馏结构

对于 BERT、RoBERTa 对应的学生模型，在通用语料库预训练，模型配置，Transformer Block 层数为 3 层，隐层维度为 1024，Tokenizer 仍然采用原教师模型使用的 Tokenizer 和词表。随机初始化参数。同样的采取了和 2.5.3 中一样的一些训练技巧，以加速训练或节省显存。进行教师模型对应的预训练任务预训练，和第 2.2 节、第 2.3 节的基本一致。

而对于 XLM，略微不同之处在于，为了简单考虑，未进行 TLM 任务的预训练。原因是该任务的数据是有监督的，需要使用两种语种翻译对齐的两句语料库，较为麻烦。作为模型压缩部署实验，目的是为了对比验证蒸馏后的模型最终在下游 TCC 任务的表现，该区别或许会造成在 XLM 评估基准上的下降，但由于后面还有两阶段蒸馏的持续学习，预计该处预训练不一致不会造成太大影响。

然后再使用教师模型指导学生模型在通用语料库上蒸馏，以及在下游 TCC 任务上蒸馏，采取事先通过教师模型统一生成指导信号给通用的语料标注，然后训练学生模型，损失函数加上指导信号的损失，在任务上蒸馏时，损失函数多加上一个真实标注的损失。

4.4 本章小结

本章介绍了当前大规模预训练模型的部署问题，常用的压缩方案，并分析了在当前的下游 TCC 任务上的适应性，得出蒸馏最合适的结论，并参考第 3 章继续预训练的思想，提出了多阶段蒸馏的策略，和一个浅而宽的 Transformer 结构的学生模型，以及具体蒸馏训练步骤。

5 实验结果及评估

本文中第二章的实验目标是验证对不同预训练语言模型添加下游网络结构改进对 TCC 任务的提升效果，第三章的实验目标是验证轻量 TAPT 对 TCC 任务有效性及最低资源的参数设置，第四章的实验目标是验证模型蒸馏后在 TCC 任务中的推断速度提升效果及损失的性能程度。本章介绍第二、三、四章中使用的数据集、实验环境以及各实验结果分析，以及存在的局限性。

5.1 TCC 任务数据集

本文下游网络结构改进微调、轻量 TAPT、多阶段蒸馏实验，选择了四项 TCC 任务来评估。在选择 TCC 任务时，研究的目标是覆盖不同的社交媒体平台、数据集大小和分类类型，如表 5-1 所示。四个选定的数据集包含从 14998 到 159571 条评论，其中数据集 Founta[62]的原始版本包括 80000 多条有 ID 发布的推文，已经成功检索到 50425 条有效推文，其余缺失的推文由于被删除而未能被检索到，涵盖三个不同的社交媒体平台和不同的分类类型（多类和多标签）。

表 5-1 四项毒性评论分类任务

数据集	数据来源	数据量	标签	分类任务
Kumar[63]	Facebook	14998	无攻击（42%），明显攻击（35%），隐性攻击（23%）	多类
Waseem[64]	Twitter	18625	种族主义（11%），性别主义（20%），都有（69%），都无	多类
Founta[62]	Twitter	50425	辱骂（8%），仇恨（3%），正常（73%），垃圾信息（16%）	多类
Wiki[1]	Wikipedia	159571	毒性（10%），重度毒性（1%），下流（5%），威胁（0.3%），侮辱（5%），身份憎恨（1%）	多标签

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

5.2 实验环境

（1）预处理

将每个输入实例（即一段评论）的最大长度限制为 512 个 tokens。本文中涉及预训练、微调、蒸馏三种训练方式，在前面各章节中有对应的数据处理具体介绍。

（2）超参数

对于本工作中的全部有监督学习任务，即在 TCC 任务上微调训练，以及下游蒸馏，使用 16 的批量大小对模型训练 3 个轮次。对于所有的无监督学习任务，即轻量 TAPT 的第一阶段，按照第 3.4 节所述，使用 16 的批量大小，对模型训练不同的轮次。所有的训练过程都使用 AdamW 优化器和 5×10^{-5} 的学习率。这与 BERT[20]建议的超参数设置略有不同，在第 2.3 节有详细说明，使得在下游任务中表现良好。所有多类分类任务都使用 Softmax 作为最终激活函数，多标签分类任务则使用 Sigmoid。

（3）环境

在所有的实验中使用双卡的 Tesla V100-SXM2-16GB GPU。与 SciBERT[23]（8 个内核的单卡 TPU v3）和 TAPT [25]（Google Cloud v3-8 TPU）中描述的语言模型领域内微调相比，硬件要求没有那么高。这使得结果更容易复现。代码实现使用 HuggingFace Transformers 4.15.0 库和 Pytorch 1.9.0。

具体依赖库环境如表 5-2 所示。

表 5-2 环境依赖

依赖 python 库	版本
harvesttext	0.8.1.4
loguru	0.5.3
numpy	1.19.2
pyhanlp	0.1.79
scikit_learn	1.0.2
torch	1.9.0
tqdm	4.50.2
transformers	4.15.0
tensorboard	2.7.0

5.3 结果及分析

第 2.5 节中为每个预训练的语言模型建立一个基线模型，并尽可能使用最简单的组件。因此，每个基线模型都采用一个预训练的语言模型，不需要继续预训练，并在其上添加一个线性分类结构。

为了尽可能控制模型参数在同一规模比较，将每个预训练的语言模型选择参数范围在 1.1 亿到 1.44 亿之间的版本。这样一来，三个模型就有了近似的相同规模。另外，这些规模比大多数同类模型的版本要小，所以它们的训练速度也比较快。使用以下版本的 BERT、RoBERTa 和 XLM：bert-base-cased（12 层，768 隐层维度，12 头，1.1 亿参数）；roberta-base（12 层，768 隐层维度，12 头，1.25 亿参数）；xlm-mlm-enfr-1024（6 层，1024 隐层维度，8 头，1.44 亿参数）。

5.3.1 下游网络结构的影响

表 5-3 不同的语言模型及添加不同的下游网络结构微调获得的 F1 分数（微观和宏观）

模型	**Kumar**(15k)		**Waseem**(18.6k)		**Founta**(50.4k)		**Wiki**(159.6k)	
	微观 F1	宏观 F1	微观 F1	宏观 F1	微观 F1	宏观 F1	微观 F1	宏观 F1
BERT-[CLS]	0.5807	0.5732	0.8722	0.6056	0.7772	0.5982	0.7724	0.6029
BERT-Linear	**0.5904**	**0.5805**	**0.8738**	**0.6196**	**0.7957**	**0.6071**	**0.7816**	**0.6372**
BERT-CNN	0.5796	0.5665	0.8480	0.5971	0.7888	0.5929	0.7388	0.4800
BERT-BiLSTM	0.5750	0.5611	0.8469	0.5921	0.7929	0.5959	0.7504	0.5945
RoBERTa-[CLS]	0.5507	0.5132	0.8522	0.6016	0.7772	0.6282	0.7724	0.6419
RoBERTa-Linear	**0.5667**	**0.5441**	**0.8673**	**0.6159**	**0.8025**	**0.6315**	**0.7822**	**0.6510**
RoBERTa-CNN	0.5313	0.4819	0.8491	0.5902	0.7892	0.5367	0.7357	0.4658
RoBERTa-BiLSTM	0.5450	0.5013	0.8426	0.5828	0.7769	0.5563	0.7405	0.5431
XLM-[CLS]	0.5607	0.5433	0.8412	0.5731	0.7972	0.6082	0.7424	0.4919
XLM-Linear	**0.5646**	**0.5503**	**0.8464**	**0.5957**	0.7989	**0.6149**	**0.7594**	**0.5122**
XLM-CNN	0.5525	0.5345	0.8362	0.5860	**0.8023**	0.6015	0.7233	0.4642
XLM-BiLSTM	0.5600	0.5409	0.8346	0.5794	0.7987	0.5751	0.7519	0.4690

如表 5-3 所示，比较不同下游网络结构的结果，每个预训练的语言模型在每项任

务上的最佳表现用黑体字显示。“BERT”，“RoBERTa”和“XLM”代表了每个预训练语言模型的基线模型，而“BERT-CNN”则作为一个例子，表示预训练语言模型添加一个 CNN 结构作为下游网络结构。

首先比较了每个模型的宏观 F1 分数，因为与微观 F1 分数相比[104]，它突出了一个模型在样本不均衡的数据集上处理少数类样本的能力。在实际情况当中，大多数 TCC 任务通常处理的都是高度不平衡的数据，其结果是，当只看微观 F1 分数时，模型在少数类（即毒性评论）上的表现往往被多数类掩盖了[104]。

如表 5-3 所示，宏观 F1 分数结果表明，在使用 CNN 或 BiLSTM 作为下游网络结构的情况下，表现始终比仅用[CLS]Token 直接分类和仅添加线性层要差。就微观 F1 而言，基线模型在所有数据集上都获得了最高的 F1 分数，但 XLM 在 Founta 数据集上除外。而在较小的数据集，即 Kumar 和 Eveem，以及多标签分类任务（Wiki），与 CNN 和 BiLSTM 下游网络结构相比的差异最大。比如，与多标签分类任务（Wiki）和最小数据集（Kumar）的基线模型相比，RoBERTa-CNN 的微观 F1 分数分别减少了 0.0465 和 0.0354，并且在 Founta 上减少了 0.0133，其数据集是多类的，比 Kumar 大三倍。在 BERT 组和 XLM 组的实验结果上也观察到类似的情况。

比较 CNN 和 BiLSTM 作为预训练语言模型的下游结构，没有一致的结果表明哪一个更好。以上结果表明，在 TCC 任务的背景下，将预先训练好的语言模型迁移到复杂的下游网络结构中，并没有比简单的线性结构多带来任何好处，并且直接使用[CLS]Token 的输出用于分类效果，比用上句子全部位置的输出接一个线性层的效果要差。其中一个可能的原因是，复杂的 DNN 结构可能会冲淡或干扰语言模型从对极其庞大的语料库进行预训练中学到的一般表示。得益于基于注意力机制的 Transformer 结构，简单而有效，预训练语言模型在大规模的自然语料上自监督预训练，能够很好的自下而上的建模语法及语义表示，已经大大超越了传统的 CNN 和 Bi-LSTM。经过多层 Transformer 结构，上层的输出近乎线性可分，也就无需复杂的 CNN 建模捕获局部特征，以及 Bi-LSTM 建模序列语义了，并且[CLS]Token 并不如加线性层。

5.3.2 领域内继续预训练的影响

综上所述，第 3.4 节在四个 TCC 任务上使用三个公开预训练语言模型，训练六个不同的轮次，进行了领域内继续预训练的测试。如表 5-3 中所示，报告中总共记录了 72 个模型的对比实验结果。其中“BERT-e1”是指对 BERT 语言模型进行 1 个轮次预训练，然后将其迁移到目标 TCC 任务上微调。每个预训练的语言模型在每个任务上的最佳结果都以黑体字标出，其结果比基线模型差的模型用斜体表示。

首先，如表 5-4 所示，与基线模型相比，继续在目标任务上对模型进行预训练往往能进一步提高性能。但在使用 BERT 和 RoBERTa 时，在 Founta 数据集和 Wiki 数据集上例外。特别是在较小的数据集上，Kumar 和 Waseem，通过轻量 TAPT 获得的性能改进是明显的。例如，对于 BERT、RoBERTa 和 XLM，当使用未标注的任务数据对语言模型进行 50 轮次的继续预训练时，在 Waseem 上的宏观 F1 分数别增加了 0.1994、0.2046 和 0.1427。比微观 F1 更高的增幅也表明，继续的预训练特别有利于数据集中的少数类，而这些往往是毒性评论的类，而不是无毒评论的类。而对于 Wiki 这个四种数据集中最大的数据集，BERT 和 RoBERTa 的基线模型取得了强大的性能，在该领域内的继续预训练对其分类性能表现出了不利的影响，唯一的例外是 XLM。三个继续预训练的语言模型在 Founta 任务中没有明显的优势或劣势。

另一个发现是，在四项任务中，哪一个训练轮次的设置是最好的，没有得到一个一致的结论。然而，对于多类分类任务 Kumar、Waseem 和 Founta，模型预训练 10 或 20 个训练轮次获得了最高的 F1 分数，与其使用较多的训练轮次的同类模型的结果相当。

综上所述，对于多标签分类任务 Wiki，对预训练语言模型进行领域内继续预训练并没有表现出明显的优势。这个任务的数据集至少要比其他三个任务的数据集大三倍。因此，实验结果表明，在相对较小的数据集上，在领域内的继续预训练对 TCC 任务是有益的。这可能是因为一个大的数据集可能已经为模型的学习提供了足够的信息；因此，来自于预训练的语言模型的特征只能提供更少的价值来学习。本质上，由于预训练的语言模型是在大规模的通用语料库上训练的，其分布和下游任务领域内的语料分布上存在差异，导致对领域内语料的特定语义和语法表示不足，以及词

表的不同，前面第 2.5.3 小节提到，当被切分成的子词在词表里找不到，即 OOV 问题，都影响后续任务的表现。此外，对于数据有限的 TCC 任务，更多的训练轮次不

表 5-4 比较基线模型的 F1 分数

模型	**Kumar**(15k)		**Waseem**(18.6k)		**Founta**(50.4k)		**Wiki**(159.6k)	
	微观 F1	宏观 F1	微观 F1	宏观 F1	微观 F1	宏观 F1	微观 F1	宏观 F1
BERT	0.5904	0.5805	0.8738	0.6196	0.7957	0.6071	**0.7816**	0.6372
BERT-e1	0.6050	0.5938	0.8737	0.6196	**0.7981**	0.6096	*0.7422*	*0.5462*
BERT-e5	0.6004	0.5899	0.8781	0.6198	0.7965	**0.6185**	*0.7475*	*0.5648*
BERT-e10	0.6029	0.5894	**0.8824**	0.7538	0.7957	0.6076	*0.7804*	**0.6539**
BERT-e20	0.6058	0.5926	0.8743	0.8176	0.7959	0.6079	*0.7404*	*0.6206*
BERT-e50	**0.6092**	**0.5972**	*0.8727*	**0.8190**	*0.7914*	*0.6025*	*0.7426*	*0.5581*
BERT-e100	0.6046	0.5898	0.8765	0.7479	*0.7906*	*0.6046*	*0.7395*	*0.5196*
RoBERTa	0.5667	0.5441	0.8673	0.6159	0.8025	0.6315	**0.7822**	**0.6510**
RoBERTa-e1	0.6075	0.5924	0.8690	0.6160	**0.8106**	0.6382	*0.7393*	*0.5067*
RoBERTa-e5	0.6075	0.5961	**0.8808**	0.6277	*0.8019*	**0.6391**	*0.7607*	*0.5650*
RoBERTa-e10	0.6146	**0.6051**	0.8711	0.6174	0.8060	0.6368	*0.7530*	*0.5606*
RoBERTa-e20	0.6063	0.5922	0.8711	*0.6134*	*0.8011*	*0.6275*	*0.7736*	*0.5875*
RoBERTa-e50	0.6146	0.6045	0.8759	**0.8205**	*0.7997*	*0.6163*	*0.7711*	*0.6503*
RoBERTa-e100	**0.6167**	0.6043	0.8743	0.7439	0.8027	*0.6242*	*0.7683*	*0.6209*
XLM	0.5646	0.5503	0.8464	0.5957	0.7989	0.6149	0.7594	0.5122
XLM-e1	0.5754	0.5566	0.8693	0.6063	**0.8044**	**0.6248**	0.7548	0.6247
XLM-e5	0.5883	0.5762	0.8593	0.6057	*0.7987*	0.6184	0.7600	0.5837
XLM-e10	0.6000	0.5896	0.8636	0.7364	0.8003	0.6170	0.7661	0.5952
XLM-e20	0.5988	0.5891	**0.8695**	**0.7826**	*0.7987*	0.6184	0.7712	0.6006
XLM-e50	0.6158	0.6091	0.8690	0.7384	0.7999	0.6178	0.7707	**0.6427**
XLM-e100	**0.6196**	**0.6097**	0.8727	0.7883	*0.7985*	0.6155	**0.7748**	0.6233

一定会带来性能上的好处。另外，在这种情况下，较小批量大小例如 16，也是可行的。一个可能的解释是：过多的训练轮次可能回导致灾难性遗忘，即在后续的微调期间，会使得在大规模语料库预训练的语言模型中获得的有用特征被遗忘，参数权重往过于学习下游数据集的分布方向更新，从而导致模型的性能下降[42]。虽然实验

中没有直接与 TAPT 进行比较，但这工作对未来的研究者来说仍然是有实际意义的。因为完全复制 TAPT 对硬件资源的要求非常高，大多数人在实际情况下无法获得。实验的结果表明，即使超参数设置的很低，在领域内继续预训练仍然有助于 TCC 任务，特别是当 TCC 的训练数据集相对较小且不平衡时。

5.3.3 不同的预训练语言模型的影响

如表 5-3 所示，当应用相同的下游结构时，BERT 的表现普遍优于 RoBERTa 和 XLM。在比较他们的宏观 F1 分数时，特别是在像 Kumar 这样的小数据集上，这个结论显得更加清楚。例如，BERT-CNN 在 Kumar 上获得的宏观 F1 分数为 0.5665，明显高于 RoBERTa（0.4819）和 XLM（0.5345）。

随着在领域内的继续预训练，RoBERTa 的表现普遍优于 BERT 和 XLM，特别是在每个 TCC 任务上比较它们的最佳轻量 TAPT 模型和基线模型的宏观 F1 分数时（结果见表 5-4）。此外，相比较 RoBERTa 和 BERT，XLM 更受益于轻量 TAPT，特别是在具有相对较小数据集的 TCC 任务上。这可以观察到，所有轻量 TAPT 的 XLM 模型在宏观 F1 分数上的表现都超过了其 XLM 的基线模型。

上述结果背后的一个可能的原因是，XLM 在其预训练中会处理长句子或文件（4000 个由句子组成的标记），而 TCC 通常处理短句子和文本[22]。因此，轻量 TAPT 使用领域内数据对语言模型进行微调，可以帮助 XLM 学习短句文本的表示。另一方面，BERT 和 RoBERTa 是在最大序列长度为 512 个 Tokens 的语料库上预训练的，这与许多 TCC 数据集相似。这可能解释了为什么它们的表现普遍优于 XLM，而且从轻量 TAPT 中受益较少。

5.3.4 多阶段模型蒸馏的影响

如表 5-5 所示，BERT-Tiny 是指 BERT 模型进行蒸馏得到的学生模型，“RoBERTa-Tiny”是指 RoBERTa 模型进行蒸馏得到的学生模型，“XLM-Tiny”是指 XLM 模型进行蒸馏得到的学生模型然后将其迁移到目标 TCC 任务上。

其中 BERT 和 RoBERTa 蒸馏后，推断速度均提升了 9.4 倍；BERT 效果损失在 3%以内；不过 XLM 在 Kumar 的宏观 F1，效果下降了较多，下降了 5%，在 Waseem

的宏观 F1，下降了 19%，在 Wiki 的宏观 F1，下降了 13%。这是由于多阶段的持续学习带来的不稳定的累计噪声，以及在前面第 4.3.3 小节提到，为了简单考虑，XLM 对应的学生模型的第一阶段预训练与第二阶段的蒸馏，是与原 XLM 不一致的，少了 TLM 任务。另外一个可能的原因在于，多阶段蒸馏思想上是持续学习，切换不同的训练任务的策略会导致之前学到的知识丢失，被称为灾难性遗忘。

表 5-5 比较基线模型的 F1 分数

模型	推断速度	**Kumar**(15k)		**Waseem**(18.6k)		**Founta**(50.4k)		**Wiki**(159.6k)	
		微观 F1	宏观 F1	微观 F1	宏观 F1	微观 F1	宏观 F1	微观 F1	宏观 F1
BERT	**1x**	**0.6092**	**0.5972**	**0.8824**	**0.8190**	**0.7981**	**0.6185**	**0.7816**	**0.6539**
BERT-Tiny	**9.4x**	*0.5950*	*0.5826*	*0.8765*	*0.7579*	*0.7905*	*0.6046*	*0.7595*	*0.6206*
RoBERTa	**1x**	**0.6167**	**0.6051**	**0.8808**	**0.8205**	**0.8106**	**0.6391**	**0.7822**	**0.6510**
RoBERTa-Tiny	**9.4x**	0.6063	0.5922	0.8690	*0.8134*	*0.7997*	*0.6163*	*0.7593*	*0.6050*
XLM	**1x**	**0.6196**	**0.6097**	**0.8695**	**0.7826**	**0.8044**	**0.6248**	**0.7748**	**0.6427**
XLM-Tiny	**9.2x**	0.5883	0.5566	0.8693	0.5957	*0.7985*	0.6155	0.7548	0.5122

总体来说，对于 BERT 和 RoBERTa 蒸馏使用浅而宽的 3 层 Transformer Block 和 1024 维隐层维度的结构作为学生模型。实现了在下游 TCC 任务中，较少的性能损失，和极大的推断速度提升，以及 3 倍的参数规模的下降，极大地减少了对部署环境的显存和计算性能的要求。

5.4 实验局限性

由于受到计算资源的限制，本研究工作缺乏一套与 TAPT 相同的参数设置的实验，这可能会进一步说明轻量 TAPT 在 TCC 任务上与原来的 TAPT 相比的情况。此外，由于 TAPT 中使用的原始数据集不可用，这里原始数据集指的是 REVIEW 领域的两个数据集，这是一个与 TCC 任务相对类似的领域，使得无法在相同的数据集上直接比较轻量 TAPT 和 TAPT。另一个未探索的因素是，每个语言模型和复杂的结构，如 CNN 和 BiLSTM，都可能有对自己有利的超参数设置。然而，在本文的研究中，

这些超参数的设置是根据以往研究中的常见设置统一的，但没有进一步精细化调参。另外，这些超参数设置对性能的影响程度还不清楚。

另外，对于 XLM 这个多语言的模型，数据集的准备需要更精细的按语种多项式采样，但是尚未评估通用领域语料库的分布情况，难以达到原论文额外补充的多语种数据，尚未能最大程度发挥其性能。

在模型部署的实验中，对模型的加速仍然还有进一步的优化空间。比如其它的工作，从底层框架入手，将模型格式转化，通过算子融合，动态图转静态图的方案，摆脱对 python 环境、pytorch 框架的依赖，转化成 C++部署。

5.5 本章小结

本章首先介绍了本文各实验使用的下游 TCC 任务数据集，然后给出实验环境细节，和各实验结果及分析，最后对于实验表现不佳之处也指出了实验的局限性。

6 总结与展望

6.1 本文主要工作

本文研究了如何更好地利用基于预训练语言模型的方法进行毒性评论分类。论文的主要工作总结如下：

（1）提出简单线性层作为三种预训练语言模型下游网络结构并使用整句输入的策略，证明优于使用 CNN、BiLSTM 复杂结构和原始只使用[CLS]的方法，能更好地提升 TCC 任务效果。

（2）提出先利用任务无标注数据对基础预训练模型部分继续预训练的轻量 TAPT 策略，及对应最低资源参数，证明其能够对 TCC 任务效果进一步提升。

（3）提出每种预训练模型对应的 Tiny 学生模型，及多阶段模型蒸馏策略，证明能够在损失较少性能的情况下大幅提升模型推断速度，减少参数规模。

6.2 展望

未来的工作将探索几个方向。例如，在下游任务的不同维度探索这些研究，看看本文的实验结论是否在这些方面有很好的泛化性。

了解到NLP最新的工作Prompt新的预训练范式，有超越预训练微调范式的势头，值得在 TCC 任务上与预训练微调范式的对比。以及一些有潜力的下一代 Attention 设计——门控线性单元，作为更高效的 Transformer 设计。

研究预训练的语言模型中将某些层的参数的系统化解冻和冻结方法。以及在根据实际硬件资源和推理速度要求的场景下，尝试从量化、裁剪、底层算子融合，转化格式，将模型部署在更低资源的 CPU 环境下的一些尝试。

参考文献

[1] 刘峰. 论消除网络暴力对构建和谐社会的重要作用. 信息网络安全，2010(10): 35-36

[2] 王乐，张紫琼，崔雪莹. 虚假评论的识别与过滤：现状与展望. 电子科技大学学报(社科版)，2022,24(01): 31-41+64

[3] 张杰，毛艺融. 平台型媒体内容审核：动因、现状与突破. 出版科学，2021,29(06): 76-83

[4] 李丹丹. 网络平台内容审查范围的界定. 信息安全研究，2019,5(09): 834-842

[5] Zeerak Waseem. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In: Proceedings of the first workshop on NLP and computational social science, Austin, TX, USA, November 5, 2016, Association for Computational Linguistics, 2016: 138-142

[6] Zeerak Waseem, Thomas Davidson, Dana Warmsley, et al. Understanding abuse: A typology of abusive language detection subtasks. In: Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, August 4, 2017, Association for Computational Linguistics, 2017: 78-84

[7] 黄杰. 自然语言处理在文本审核中的应用. 网络安全技术与应用，2021(03): 27-29

[8] Pete Burnap, Omer F. Rana, Nick Avis, et al. Detecting tension in online communities with computational Twitter analysis. Technological Forecasting and Social Change, 2015, 95: 96-108

[9] Vikas S. Chavan, Shylaja S. S. Machine learning approach for detection of cyber-aggressive comments by peers on social media network. In: 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, Aug 10-13, 2015, IEEE, 2015: 2354-2358

[10] Irene Kwok, Yuzhou Wang. Locate the hate: Detecting tweets against blacks. In: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI'13), Bellevue, Washington, July 14-18, 2013, AAAI Press, 2013: 1621-1622

[11] Paula Fortuna, Sérgio Nunes. A survey on automatic detection of hate speech in text.

ACM Computing Surveys (CSUR), 2018, 51(4): 1-30

[12] Ji H. Park, Pascale Fung. One-step and two-step classification for abusive language detection on twitter. In: Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, August 4, 2017, Association for Computational Linguistics, 2017: 41-45

[13] Zeerak Waseem, Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In: Proceedings of the NAACL student research workshop, San Diego, California, June 12-17, 2016, Association for Computational Linguistics, 2016: 88-93

[14] Betty v. Aken, Julian Risch, Ralf Krestel, et al. Challenges for toxic comment classification: An in-depth error analysis. In: Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), Brussels, Belgium, October 31, 2018, Association for Computational Linguistics, 2018: 33-42

[15] Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, et al. Dynamic Multi-Task Learning with Convolutional Neural Network. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17), Melbourne, Australia, August 19-25, 2017, AAAI Press, 2017: 1668-1674

[16] Han Liu, Pete Burnap, Wafa Alorainy, et al. Fuzzy multi-task learning for hate speech type identification. In: The World Wide Web Conference (WWW '19), San Francisco, CA, USA, May 13-17, 2019, Association for Computing Machinery, 2019: 3006-3012

[17] Anna Schmidt, Michael Wiegand. A survey on hate speech detection using natural language processing. In: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, Valencia, Spain, April 3, 2017, Association for Computational Linguistics, 2019: 1-10

[18] Fabio D. Vigna, Andrea Cimino, Felice dell'Orletta, et al. Hate me, hate me not: Hate speech detection on facebook. In: Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, January 17-20, 2017: 86-95

[19] Ziqi Zhang, David Robinson, Jonathan A. Tepper. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In: The Semantic Web - 15th International Conference (ESWC 2018), Heraklion, Crete, Greece, June 3-7,

2018, Springer, 2018: 745-760

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), Minneapolis, MN, USA, June 2-7, 2019, Association for Computational Linguistics, 2019: 4174-4186

[21] Alexis Conneau, Guillaume Lample. Cross-lingual language model pretraining. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS 2019), Vancouver, BC, Canada, December 8-14, 2019: 7057-7067

[22] Yinhan Liu, Myle Ott, Naman Goyal, et al. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv: 1907.11692, 2019

[23] Iz Beltagy, Kyle Lo, Arman Cohan. SciBERT: A pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), Hong Kong, China, November 3-7, 2019, Association for Computational Linguistics, 2019: 3613-3618

[24] Alexandra Chronopoulou, Christos Baziotis, Alexandros Potamianos. An Embarrassingly Simple Approach for Transfer Learning from Pretrained Language Models. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), Minneapolis, MN, USA, June 2-7, 2019, Association for Computational Linguistics, 2019: 2089-2095

[25] Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, et al. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020), Online, July 5-10, 2020, Association for Computational Linguistics, 2020: 8342-8360

[26] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, et al. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 2020, 36(4): 1234-1240

[27] Manish Munikar, Sushil Shakya, Aakash Shrestha. Fine-grained Sentiment

Classification using BERT. In: 2019 Artificial Intelligence for Transforming Business and Society (AITB), Kathmandu, Nepal, November 5-5, 2019, IEEE, 2019, 1: 1-5

[28] Marzieh Mozafari, Reza Farahbakhsh, Noël Crespi. A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. In: Proceedings of the Eighth International Conference on Complex Networks and Their Applications (COMPLEX NETWORKS 2019), Lisbon, Portugal, December 10-12, 2019, Springer, 2019: 928-940

[29] Zhengjie Gao, Ao Feng, Xinyu Song, et al. Target-Dependent Sentiment Classification With BERT. IEEE Access, 2019, 7: 154290-154299

[30] Matthew Tang, Priyanka Gandhi, Md A. Kabir, et al. Progress notes classification and keyword extraction using attention-based deep learning models with BERT. arXiv preprint arXiv: 1910.05786, 2019

[31] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, et al. Deep Learning for Hate Speech Detection in Tweets. In: Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017, ACM, 2017: 759-760

[32] Thomas Davidson, Dana Warmsley, Michael W. Macy, et al. Automated Hate Speech Detection and the Problem of Offensive Language. In: Proceedings of the Eleventh International Conference on Web and Social Media (ICWSM 2017), Montréal, Québec, Canada, May 15-18, 2017, AAAI Press, 2017: 512-515

[33] Vinita Nahar, Sanad Al-Maskari, Xue Li, et al. Semi-supervised Learning for Cyberbullying Detection in Social Networks. In: Databases Theory and Applications - 25th Australasian Database Conference (ADC 2014), Brisbane, QLD, Australia, July 14-16, 2014, Springer, 2014: 160-171

[34] Andrew M. Dai, Quoc V. Le. Semi-supervised Sequence Learning. Advances in neural information processing systems, 2015, 28: 3079-3087

[35] Matthew E. Peters, Mark Neumann, Mohit Iyyer, et al. Deep Contextualized Word Representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018), New Orleans, Louisiana, USA, June 1-6, 2018, Association for Computational Linguistics, 2018: 2227-2237

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, et al. Improving language

understanding with unsupervised learning. Technical report, OpenAI, 2018: 4

[37] Jeremy Howard, Sebastian Ruder. Universal language model fine-tuning for text classification. arXiv preprint arXiv: 1801.06146, 2018

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv: 1609.08144, 2016

[39] Tomas Mikolov, Kai Chen, Greg Corrado, et al. Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781, 2013

[40] Wilson L. Taylor. Cloze Procedure: A New Tool for Measuring Readability. Journalism and Mass Communication Quarterly, 1953, 30(4): 415

[41] Yukun Zhu, Jamie Kiros, Richard Zemel, et al. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In: 2015 IEEE International Conference on Computer Vision (ICCV 2015), Santiago, Chile, December 7-13, 2015, IEEE, 2015: 19-27

[42] Ciprian Chelba, Tomas Mikolov, Mike Schuster, et al. One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint arXiv: 1312.3005, 2013

[43] Ankur P Parikh, Oscar Täckström, Dipanjan Das, et al. A decomposable attention model for natural language inference. arXiv preprint arXiv: 1606.01933, 2016

[44] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, et al. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv: 1611.01603, 2016

[45] Alex Wang, Amanpreet Singh, Julian Michael, et al. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Brussels, Belgium, November 1, 2018, Association for Computational Linguistics, 2018: 353-355

[46] 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, Austin, Texas, USA, November 1-5, 2016, Association for Computational Linguistics, 2016: 2383-2392

[47] Guokun Lai, Qizhe Xie, Hanxiao Liu, et al. RACE: Large-scale ReAding

Comprehension Dataset From Examinations. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017), Copenhagen, Denmark, September 9-11, 2017, Association for Computational Linguistics, 2017: 796-805

[48] Yang You, Jing Li, Jonathan Hseu, et al. Reducing BERT pre-training time from 3 days to 76 minutes. arXiv preprint arXiv: 1904.00962, 2019

[49] Rico Sennrich, Barry Haddow, Alexandra Birch. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909, 2015

[50] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, et al. Unsupervised machine translation using monolingual corpora only. arXiv preprint arXiv: 1711.00043, 2017

[51] Samuel L. Smith, David H. P. Turban, Steven Hamblin, et al. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. arXiv preprint arXiv: 1702.03859, 2017

[52] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, et al. Exploring the limits of language modeling. arXiv preprint arXiv: 1602.02410, 2016

[53] Zihang Dai, Zhilin Yang, Yiming Yang, et al. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In: Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL 2019), Florence, Italy, July 28-August 2, 2019, Association for Computational Linguistics, 2019: 2978-2988

[54] Paul J. Werbos. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 1990, 78(10): 1550-1560

[55] Rami Al-Rfou, Dokook Choe, Noah Constant, et al. Character-level language modeling with deeper self-attention. In: Proceedings of the AAAI conference on artificial intelligence, Honolulu, Hawaii, USA, January 27 - February 1, 2019, AAAI Press, 2019: 3159-3166

[56] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), Doha, Qatar, October 25-29, 2014, ACL, 2014: 1746-1751

[57] Alex Graves, Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural networks, 2005,

18(5-6): 602-610

[58] Colin Raffel, Noam Shazeer, Adam Roberts, et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 2020, 21: 1-67

[59] Zhilin Yang, Zihang Dai, Yiming Yang, et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding. Advances in neural information processing systems, 2019, 32

[60] Alex Wang, Yada Pruksachatkun, Nikita Nangia, et al. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. Advances in neural information processing systems, 2019, 32

[61] Luciano Floridi, Massimo Chiriatti. GPT-3: Its nature, scope, limits, and consequences. Minds and Machines, 2020, 30(4): 681-694

[62] Paula Fortuna, Sérgio Nunes. A survey on automatic detection of hate speech in text. ACM Computing Surveys (CSUR), 2018, 51(4): 1-30

[63] Ritesh Kumar, Aishwarya N. Reganti, Akshit Bhatia, et al. Aggression-annotated corpus of hindi-english code-mixed data. arXiv preprint arXiv: 1803.09402, 2018

[64] Zeerak Waseem. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. NLP+ CSS 2016, 2016: 138