文章编号：2095-2163(2022)10-0075-05 中图分类号：TP391 文献标志码：A

基于 BERT 模型的指令集多标签分类研究

王淳睿[1]，何先波[2]，易 洋[1]
（1 西华师范大学 电子信息工程学院，四川 南充 637009；2 西华师范大学 计算机学院，四川 南充 637009）

摘 要： 指令分类可有效识别出指令的类别并提取出指令中的关键信息，有助于编译器开发人员快速构建编译器后端指令相关的代码，降低时间成本。针对传统文本分类模型的不足，研究提出基于 BERT 预训练模型的指令描述文本分类方法。经过数据清洗、预处理，在 BERT 预训练模型基础上，研究构建了文本多标签分类模型，实验结果表明，本文提出的方法对指令文本有较好的分类效果。

关键词： 深度学习；BERT；多标签分类；指令集

Research on instruction set multi-label classification based on BERT model

WANG Chunrui[1], HE Xianbo[2], YI Yang[1]
(1 College of Electronics Information Engineering, China West Normal University, Nanchong Sichuan 637009, China;
2 College of Computer, China West Normal University, Nanchong Sichuan 637009, China)

【Abstract】 The instruction classification can effectively identify the category of the instruction and extract the key information in the instruction, which helps the compiler developer to quickly construct the code related to the back-end instruction of the compiler and reduce the time cost. Aiming at the shortcomings of traditional text classification models, this paper proposes an instruction description text classification method based on the BERT pre-training model. After data cleaning and preprocessing, based on the BERT pre-training model, a text multi-label classification model is researched and constructed. The experimental results show that the method proposed in this paper has a good classification effect on instruction text classification.

【Key words】 deep learning; BERT; multi-label classification; instruction set

0 引 言

研究可知，编译器作为芯片生态系统中的重要一环，对自主操作系统的研发起着非常关键的基础性作用。2021 年 4 月，龙芯推出 Loong Arch 国产自主指令集（Instruction Set Architecture，ISA），一方面折射出国内芯片生态系统的发展潜力，同时也展示出了国内芯片生态系统的应用前景。然而面对全新的 ISA、例如 Loong Arch，构建出支持 Loong Arch 架构的编译器后端需要耗费大量人力以及时间[1]。本文参考低级虚拟机（Low Level Visual Machine，LLVM）中指令描述的代码，将指令划分为 13 类，并以此为分类原则训练基于 BERT 的多标签分类模型。该模型对 Mips、ARM 指令集架构手册中指令的分类有着不错的表现，根据指令分类完成的类别可减少开发人员对编译器后端指令定义相关部分所花费的时间，从而有效提升编译器构建效率。

1 相关理论及方法

1.1 BERT 训练模型

BERT（Bidirectional Encoder Representation from Transformers）是一种为不同自然语言处理任务提供支持的预训练语言表示模型[2]。该模型基于 2017 年谷歌公司发布的 Transformer 架构，由 Transformer 的双向编码器构成。区别于 ELMo 和 GPT 等单向传统语言模型，BERT 利用掩码语言模型（masked language model，MLM）进行预训练，并且采用深层的双向 Transformer 组件来构建整个模型，最终生成能融合左右上下文信息的深层双向语言表征，增加一个输出层，可以对预训练的 BERT 模型进行微调（Fine-Tuning），为各类执行任务创建先进的模型，无需针对特定任务的架构做实质性的修改[3]。

BERT 的模型架构采用了 Transformer 模型的编码部分，模型的输入由 3 种嵌入层累加构成，具体就

基金项目：西华师范大学英才科研基金项目（17YC149）。

作者简介：王淳睿（1997-），男，硕士研究生，主要研究方向：编译器开发；何先波（1971-），男，博士，教授，硕士生导师，主要研究方向：嵌入式系统研究；易 洋（1999-），女，硕士研究生，主要研究方向：深度学习、时序测序。
通讯作者：何先波 Email：1946034057@qq.com

收稿日期：2022-05-18

是:词嵌入(Token embeddings),将各个词转换成固定维度的向量;分段嵌入(Segment embeddings),用于区分不同的 sentence;位置嵌入(Position embeddings),用于表征词语位置关系。输入的编码如图 1 所示。

BERT 处理下游任务如多标签分类任务、采用的是微调的方式,一般而言,仅需要少量的改动,即可将 BERT 与下游任务融合[5],整体流程如图 2 所示。

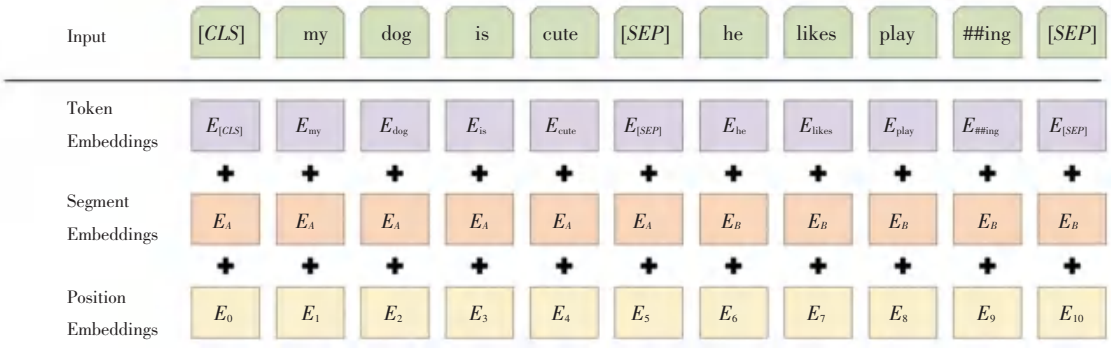

图 1 输入编码示例图[4]

Fig. 1 Input coding example diagram[4]

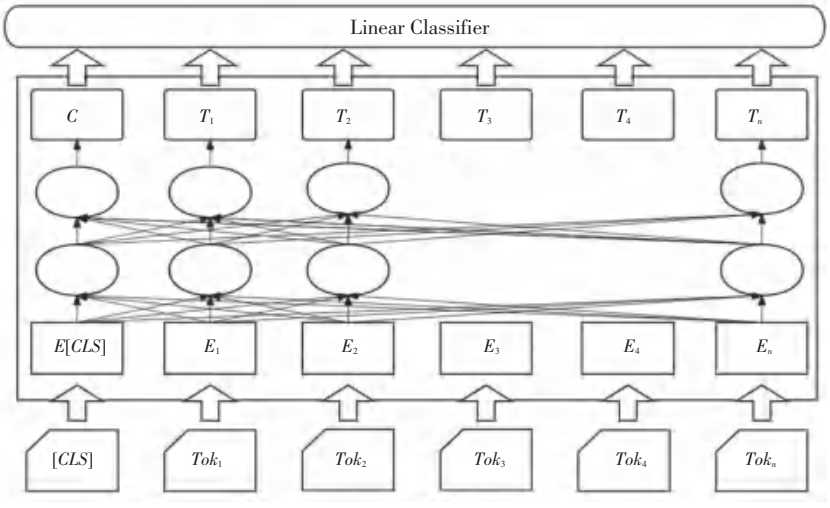

图 2 fine-tuning 架构图

Fig. 2 fine-tuning architecture diagram

作为一个预训练模型,BERT 需要应对模型的输入序列变长问题。BERT 提供了 2 种方法加以解决:

(1)使用分割 token ([*SEP*]) 插入到每个句子中,来分开不同句子 token 序列。

(2)为每个 token 特征添加一个可学习的分割 embedding 来表示句子的分割点。

区别于 ELMo 和 GPT 等单向传统语言模型,BERT 模型的优点是:

(1)采用 MLM 对双向的 Transformers 进行预训练,来生成深层的双向语言特征。

(2)在训练后,只需要额外的输出层进行微调,就能在多种任务中取得良好表现,在这个过程中不需要再对 BERT 的结构进行修改。

综合前述分析后可知,本文拟选择 BERT 作为实验的预训练模型。

1.2 指令分类原则

CPU 架构是 CPU 产品制定的规范,旨在区分不同种类的 CPU,目前市场上的 CPU 主要分为 2 类。一类是 Intel、AMD 为代表的复杂指令集 CPU,另一类是以 ARM、IBM 为代表的精简指令集 CPU。为了将不同品牌、不同类别架构的指令集进行统一的分类、从而提高构建编译器的效率,减少构建所需要的时间,本文参考了 LLVM 中描述指令的代码以及文献[6]中的方法,将指令共划分为 13 类,详见表 1。需要指出的是,目前国内外多标签分类更多的是应用在医学、新闻等领域,应用 BERT 将计算机指令进行多标签分类的研究甚少,这是本文的研究亮点之一。

表 1　指令类别划分

Tab. 1　Classification of instructions

类别	含义
isReturn	是否为返回指令
isBranch	是否为分支指令
isIndirectBranch	是否为间接分支指令
isCompare	是否为比较指令
isBarrier	是否为停止控制流之后的指令
isCall	是否为调用指令
isAdd	是否为加法指令
canFoldAsLoad	是否可以在其他指令中折叠为内存操作数
mayLoad	是否可能会读取内存
mayStore	是否可能会修改内存
isCommutable	是否为交换指令
isTerminator	是否是基本块的终止符
hasDelaySlot	是否具有代码生成器填充的延迟槽

2　相关工作

2.1　数据集制作

在已经推出的开源数据集平台中，指令集相关的数据集甚少，故本文将从 Mips32、Mips64、ARM 等指令集架构手册中手动提取出有关指令描述的文本，并且从 LLVM 的目标平台 td 文件中检索出每条指令的类别，如图 3 所示。

实验共整理 MIPS 架构指令数据 640 条，通过随机采样的方式将数据按照 8∶1∶1 的比列划分为训练集、验证集与测试集。每一个 Segment 长度均小于 512，类别包含：返回指令、条转指令、分支指令等共 13 类。和仅包含消极与积极两种类别的情感分类数据相比较，则更加清晰地展现了本文提出的模型的优越性。

由于本文的实验训练数据较少，所以增加了对 BERT 模型的预训练任务。预训练任务的数据集来自 Mips32 指令手册、ARMv8 指令手册、RISCV 指令手册等共计 16 本指令集手册，

本文首先使用训练集进行模型的训练，然后在验证集上对模型参数进行不断地调整，直至找到模型的最优参数，最后在测试集上进行测试。

Name	description	normal de	isReturn	isBranch	isCompare	isIndirectB	isBarrier	isCall	isAdd	mayLoad	mayStore	isTermina	hasDelayS	isCommut	canFoldAs
ADDW	ADDW is RV64I-only instructions that are defined analogously to ADD, but operate on 32-bit values and produce signed 32-bit results. Overflows are ignored, and the low 32-bits of the result is sign-extended to 64-bits and written to the destination register.		0	0	0	0	0	0	0	0	0	0	0	0	0

图 3　数据集示例图

Fig. 3　Dataset example graph

2.2　文本预处理

本文运用了文本的预处理方法[7]，对描述指令文本中的无效信息进行清洗，主要是对文本中的无用符号、停用词、非文本数据等进行处理。对预训练数据先将格式由 PDF 转换成 TXT 格式，再将 TXT 文件进行处理，每一行只保留一句文本，接着又将 16 本指令集手册合并为一个 TXT 格式，再将预训练数据处理成 MLM（Mask Language Module）任务的数据，如图 4 所示。

在图 4 给出的整个 MLM 任务数据集的构建流程中，第一阶段和第二阶段是根据原始语料来构造 MLM 任务所需要的输入和标签；第三阶段是随机屏蔽掉部分 *token* 来构造 MLM 任务的输入，并同时进行 padding 处理；第四阶段则是根据第三阶段处理后的结果来构造 MLM 任务的标签值，其中［*P*］表示 padding 的含义，目的是为了忽略那些不需要进行预测的 *Token* 在计算损失时的损失值[8]。

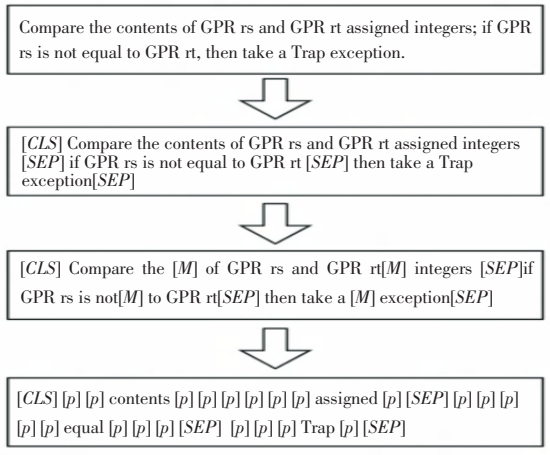

图 4　MLM 任务数据集构造流程图

Fig. 4　MLM task dataset construction flowchart

2.3　实验环境

本文实验使用 PyTorch 深度学习架构训练基于 BERT 的多标签分类模型，并配置好 GPU 在 Ubuntu16.04 系统上。实验环境配置具体见表 2。

表 2 环境配置

Tab. 2 Environmental configuration

配置内容	配置参数
开发语言	Python3.8
操作系统	Linux
显卡	GeForce RTX 2080 Ti
内存	64 G 内存 32 核

2.4 实验内容

由于本文实验训练数据的数量有限,对于目标任务数据量少的情况下如果直接训练就可能无法获得足够多的特征,如此一来则难以较好地学习到特征和一些重要信息,所以本文使用相关领域的大量数据进行预训练就可以学习到本领域的更多的特征和信息,这样迁移到数据量小的任务上进行微调就可以达到更好的效果。

BERT 预训练采用了 2 个训练任务,上文提到的 Masked LM 任务用来捕捉单词级的特征,Next Sentence Prediction 任务用来捕捉句子级的特征。本文随机屏蔽掉语料中 15%的 *token*, 然后去预测被屏蔽掉的 *token*, 将 masked token 位置的隐藏层向量输出 *softmax* 即可得到预测结果。

本文首先在预训练数据集上对 BERT 进行预训练,下游任务在训练时会将已经训练好的模型参数加载到网络中,然后使用 Mips32 数据集进行下游任务训练[9],最后找出在测试集上最优模型在其他架构数据集上进行测试,本实验详细的模型参数见表 3。

表 3 模型参数设置表

Tab. 3 Model parameters setting table

参数	值
Dropout	0.5
Batch_Size	8
Max_Length	512
Transformer 层数	12
Learning_Rate	1e-5

2.5 评价指标

本文使用 F_1 值作为评价本多标签分类实验的指标,F_1 值用于权衡 *Precision* 和 *Recall*,可定义为精确率和召回率的调和平均数。本次研究中推导得出的各数学定义的公式表述分别如下。

(1)精确率。这里用到的数学公式可写为:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

(2)召回率。这里用到的数学公式可写为:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

(3)准确率。这里用到的数学公式可写为:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

(4) F_1 值。这里用到的数学公式可写为:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

其中, *TP* (True Positive)表示预测为正,实际为正; *FP* (False Positive) 表示预测为正,实际为负; *TN* (True Negative) 表示预测为负,实际为负; *FN* (False Negative)表示预测为负,实际为负。通过统计 *TP*、*FP*、*TN*、*FN* 等数据可以计算出精确率和召回率[10]。

3 实验结果

实验结果见表 4。由表 4 可见,本文提出的多标签分类模型经过在 Mips32 训练集上训练,准确率达到了 96.4%, F_1 值达到 92.5%。分析后发现,BERT 语言模型经过训练,在 Mips32 架构指令集上有着不错的表现。再将训练后的模型使用 Mips64 和 ARMv8 架构指令集数据进行测试,准确率分别达到了 94.7%和 86.1%, F_1 值则分别达到了 90.9%与 27.9%。

表 4 实验结果

Tab. 4 Experimental results %

目标架构指令集	准确率	F_1
Mips32	96.4	92.5
Mips64	94.7	90.9
ARMv8	86.1	27.9

实验结果的柱状表示见图 5。由图 5 可以看到,模型在 Mips32 指令数据集上的准确率与 Mips64 指令数据集上的准确率相近,并且高于 ARMv8 指令数据集上的准确率,一定程度上也表明了本文提出的分类方法在同种架构中有着不错的分类表现,在 2 种不同架构的指令集上分类能力要低于同种架构指令集。

(下转第 85 页)