题目

代码预训练模型 CodeBERT:A Pre-Trained Model for Programming and Natural Languages

摘要

本次研究报告主要围绕着代码领域的CodeBERT 模型进行复现和探索。CodeBERT模型是一个双峰预训练的编程语言（PL）和 自然语言（NL）的模型。CodeBERT预训练模型能够支持下游的NL-PL任务，如代码搜索，代码文档生成等。论文使用基于transformer的神经架构开发CodeBERT，并使用混合目标函数对其进行训练，该目标函数包含替换令牌检测的预训练任务，即检测从生成器中采样的可信替代方案。这使研究人员能够利用NL-PL对的“双峰”数据和“单峰”数据，前者为模型训练提供输入令牌，而后者有助于学习更好的生成器。我们通过微调模型参数来评估两个NL-PL应用程序上的CodeBERT。论文结果表明，CodeBERT在自然语言代码搜索和代码文档生成方面都达到了最先进的性能。此外，论文作者为了研究CodeBERT学习了什么类型的知识，构建了一个用于NL-PL代码搜索的数据集，并在预训练模型参数固定的零样本学习设置中进行评估。论文结果表明，CodeBERT在NLPL探测上的性能优于先前的预训练模型。根据对论文的阅读和神经网络的搭建，本研究报告在基于原论文的基础模型上，对其在代码搜索领域进行复现，实验以及改进。

关键词：CodeBERT；自然语言处理；编程语言；预训练模型

1 引言

在当今的人工智能领域，自然语言处理（NLP）和编程语言处理是两个重要的研究方向，在其交叉领域中，如何对两者之间的进行语义对其，语义理解等等都是现今研究的热点问题。而随着大规模预训练模型的发展，如 BERT、GPT 等，其在 NLP 任务中取得了显著成果，研究人员开始探索如何将类似的技术应用于编程语言领域以及两者的结合。基于这种需求，代码预训练模型CodeBERT 作为一种创新的双模态预训练模型，旨在捕捉自然语言和编程语言之间的语义联系，为自然语言代码搜索、代码文档生成等任务提供支持，具有重要的研究价值和应用前景。本研究通过复现 CodeBERT 模型，深入了解其工作机制和性能表现，为相关领域的研究和应用提供参考和借鉴。

2 相关工作

2.1 自然语言处理中的预训练模型

在自然语言处理领域的发展进程中，过去几年无疑是大型预训练模型快速发展的关键时期。诸如 ELMo（Peters et al., 2018）、GPT（Radford et al., 2018）、BERT（Devlin et al., 2018）、XLNet（Yang et al., 2019）以及 RoBERTa（Liu et al., 2019）等模型的横空出世，极大地推动了自然语言处理任务在各个维度的发展。这些模型的核心训练范式是基于海量的大规模文本数据展开的，借助自监督学习目标来构建深度神经网络架构。其中，Transformer 架构（Vaswani et al., 2017）凭借其独特的优势，在这一领域占据了极为重要的地位并被广泛应用。

以 BERT 为例，其掩码语言建模（Masked Language Modeling，MLM）目标成为了极具影响力的创新点。在具体操作中，会从输入文本序列里随机挑选部分单词进行掩码处理，将其替换为特定的 [MASK] 标记。模型的训练任务便是依据周围未被掩码的上下文信息，全力预测这些被掩码的原始单词。这种设计使得模型能够深度挖掘文本中的语义和语法信息，从而学习到极为有效的上下文表示。在诸多自然语言处理任务中，如文本分类、情感分析、问答系统等，BERT 及其变体都取得了令人瞩目的优异成绩，为后续预训练模型的持续演进和创新奠定了坚实且不可或缺的基础。

2.2 多模态预训练模型

随着自然语言处理领域中预训练模型所取得的巨大成功，研究的浪潮迅速蔓延至多模态领域，多模态预训练模型应运而生。其核心使命在于精准探寻不同模态输入之间潜在的隐含对齐关系，进而打破模态之间的壁垒，实现信息的深度融合与协同处理。

ViLBERT（Lu et al., 2019）便是其中的典型代表之一，它聚焦于图像字幕数据这一丰富的多模态资源。在训练过程中，模型一方面致力于重建被掩码的图像区域类别或文字信息，另一方面则全力判断给定的字幕是否能够准确地描述图像的实际内容。通过这种独特的训练机制，ViLBERT 在视觉 - 语言相关任务中成功地学习到了有效的模态融合表示，并取得了一定的成果。

同样地，VideoBERT（Sun et al., 2019）将目光投向了语言 - 视频数据，通过精心设计的视频和文本掩码令牌预测任务来驱动模型的训练。在处理视频内容和与之对应的文本描述时，VideoBERT 能够有效地捕捉两者之间的语义关联，在视频理解、视频字幕生成等任务中展现出了一定的潜力。

这些多模态预训练模型的创新实践为 CodeBERT 的研究开辟了全新的思路和方向。CodeBERT 创新性地将自然语言和编程语言视作不同的模态，使用了NL-PL双模态预训练方法。它借鉴了多模态预训练模型在模态融合和训练目标设计方面的经验，同时结合自然语言和编程语言的独特特点，致力于构建一种能够有效处理两者之间复杂关系的强大模型架构，为自然语言与编程语言的交叉应用场景提供了全新的解决方案。

3 本文方法

3.1 模型架构

本研究复现的 CodeBERT 模型采用多层双向 Transformer 架构，与 BERT 和 RoBERTa 类似。具体而言，使用了与 RoBERTa - base 相同的模型结构，包含 125M 个模型参数。这种架构能够有效地捕捉输入文本的上下文信息，通过多头注意力机制对自然语言和编程语言的序列进行编码，为后续的任务处理提供强大的表示能力。

3.2 输入 / 输出表示

在预训练阶段，输入为自然语言文本和代码片段的拼接，中间用特殊的 [CLS] 标记分隔，例如：[CLS] $w_1,w_2,w_3.....w_n$ [SEP] c_1,c_2,c_3[EOS]。其中，自然语言文本被视为单词序列并采用 WordPiece 进行分词，代码则作为令牌序列处理。模型的输出包括每个令牌（自然语言和代码）的上下文向量表示以及 [CLS] 标记的表示，[CLS] 标记的最终隐藏表示用于分类或排序任务，作为整个输入序列的聚合表示。

3.3 预训练数据

复现过程中使用了来自 Github 仓库的数据，包括双模态数据（自然语言 - 代码对）和单模态数据（无配对的代码和自然语言）。具体采用了 Husain [8]等人提供的数据集，涵盖 Python、Java、JavaScript、PHP、Ruby 和 Go 六种编程语言，其中包含 2.1M 个双模态数据点和 6.4M 个单模态代码。数据经过一系列预处理，如过滤掉使用频率低的项目、截断文档、去除短函数和包含特定字符串的函数名等，以确保数据质量和训练效果。

3.4 预训练目标

CodeBERT 的预训练采用了两种目标函数：掩码语言建模（MLM）和替换令牌检测（RTD）。

• 掩码语言建模（MLM）：对于输入的自然语言 - 代码对（$(x = w, c)$），随机选择自然语言和代码中的部分位置进行掩码，用 [MASK] 标记替换，然后预测被掩码的原始令牌。通过这种方式，模型学习到基于上下文的语言表示，损失函数为

$$\mathcal{L}_{MLM}(\theta) = \sum_{i \in m^w \cup m^c} -logp^{D_1}\left(x_i | w^{masked}, c^{masked}\right)$$

，

其中 p^{D_1} 是用于预测令牌的判别器。

• 替换令牌检测（RTD）：利用大量单模态数据，通过两个生成器（自然语言生成器NL 和代码生成器Code）为随机掩码位置生成可能的替代令牌，得到 $w^{corrupt}$ 和 $c^{corrupt}$，然后训练判别器判断每个位置的令牌是否为原始令牌。这是一个二分类问题，损失函数为

$$\mathcal{L}_{RTD}(\theta) = \sum_{i=1}^{|w|+|c|} \left(\delta(i) log p^{D_2}\left(x^{corrupt}, i\right) + \left(1-\delta(i)\right)\left(1 - log p^{D_2}\left(x^{corrupt}, i\right)\right)\right)$$

其中 $\delta(i)$ 是指示函数，p^{D_2} 是判别器。最终的预训练损失函数为

$$min_\theta \mathcal{L}_{MLM}(\theta) + \mathcal{L}_{RTD}(\theta)$$

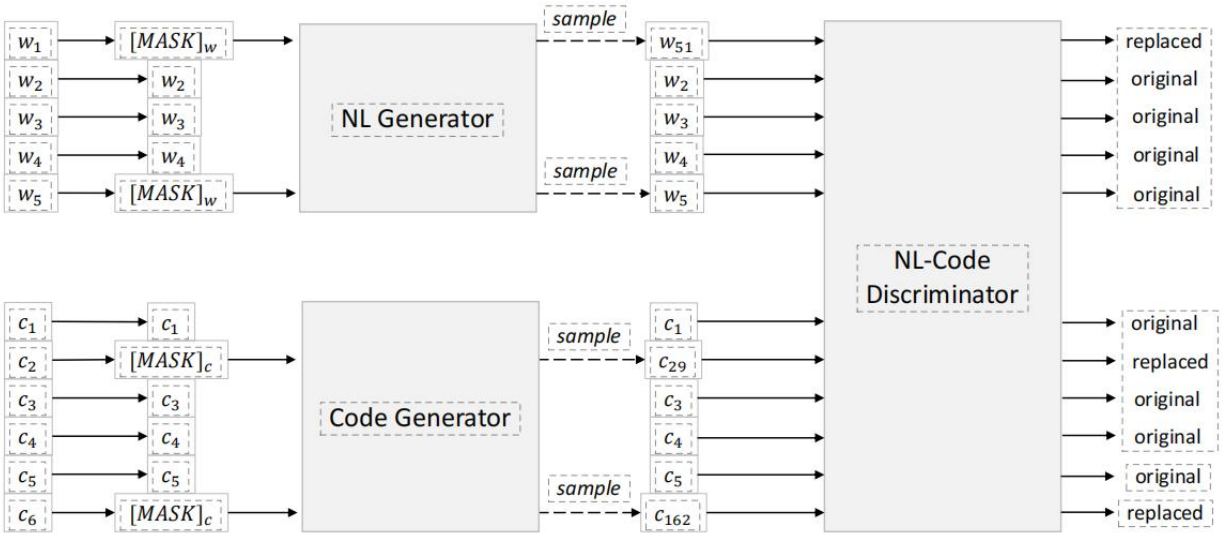

图1 RTD任务示意图

3.5 微调 CodeBERT

在下游任务中，根据不同任务设置对 CodeBERT 进行微调。例如在自然语言代码搜索任务中，输入方式与预训练阶段相同，使用 [CLS] 标记的表示来衡量代码和自然语言查询之间的语义相关性，并通过二元分类损失函数进行训练。在代码文档生成任务中，采用编码器 - 解码器框架，用 CodeBERT 初始化生成模型的编码器，然后根据具体任务需求调整模型参数。

4 复现细节

4.1 与已有开源代码对比

在复现 CodeBERT 过程中，参考了原论文作者在 Github 上提供的开源代码（https://github.com/microsoft/CodeBERT）。主要使用论文提出的预训练模型，和其数据处理方法，确保数据的预处理和模型架构的搭建符合原论文要求。同时，在训练过程中，根据实验环境对超参数的设置进行了调整，以适应本地的计算环境和实验需求。与原代码相比，本复现

工作在训练损失时进行可视化，以便于更好地理解模型的训练过程和效果；并根据现有的改进方法将其进行融合，对论文效果进行小幅度提升。

4.2 实验环境

实验环境基于一台配备4卡的3090 NVIDIA GPU 的服务器，操作系统为 Linux。使用 PyTorch 深度学习框架。具体的依赖库包括 Transformers 库用于模型的搭建和调用，以及其他相关的数据处理和可视化库，如 NumPy、Pandas 和 Matplotlib 等。通过合理配置环境变量和依赖关系，确保代码能够顺利运行。

4.3 创新点

结合数据流的输入：在模型中创新性地结合了数据流作为输入，利用数据流动态表示代码中变量间的依赖关系，使模型能更好地捕捉代码语义信息，区别于传统仅将代码视为词序列的方法，提升了模型在代码相关任务上的性能。

5 实验结果分析

在自然语言代码搜索任务中，使用 CodeSearchNet 语料库进行实验，按照原论文的评估指标计算平均倒数排名（MRR）。其中，批处理大小设置为16，原论文为64；学习率与原论文一样为0.004；最大输入tokens数为128，原论文为256。数据集构成如图2所示：

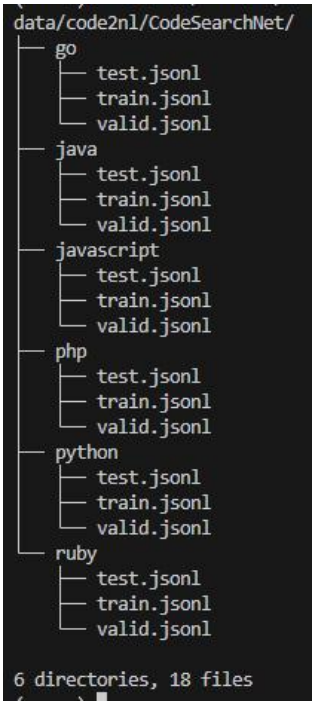

图2 数据集结构

由于此次研究是针对于CodeBert模型的复现工作，所以原论文中的用于消融实验的其他模型并未进行实验运行，故只与原论文中的CodeBert实验结果进行比较，原论文结果参考图3。

比较图3最后一行和图4中的CodeBERT相较于原论文的结果。能够看到实验后获得的准确率均有下降。因为要根据实验环境设置超参数，批训练大小减小以及最大输入tokens长度减小，因此引起了模型在代码搜索方面能力的下降，但相较于原论文的其他模型效果，还是能够证明CodeBert的有效性。

图4的第二行为加入数据流的输入，能够让模型更加关注代码结构方面。对比同实验环境下的两种模型，加入数据流输入的模型其准确率均有0.02的上升，这能够说明关注代码结构信息有益于模型对代码的理解能力。

MODEL	RUBY	JAVASCRIPT	GO	PYTHON	JAVA	PHP	MA-AVG
NBOW	0.4285	0.4607	0.6409	0.5809	0.5140	0.4835	0.5181
CNN	0.2450	0.3523	0.6274	0.5708	0.5270	0.5294	0.4753
BIRNN	0.0835	0.1530	0.4524	0.3213	0.2865	0.2512	0.2580
SELFATT	0.3651	0.4506	0.6809	0.6922	0.5866	0.6011	0.5628
ROBERTA	0.6245	0.6060	0.8204	0.8087	0.6659	0.6576	0.6972
PT W/ CODE ONLY (INIT=S)	0.5712	0.5557	0.7929	0.7855	0.6567	0.6172	0.6632
PT W/ CODE ONLY (INIT=R)	0.6612	0.6402	0.8191	0.8438	0.7213	0.6706	0.7260
CODEBERT (MLM, INIT=S)	0.5695	0.6029	0.8304	0.8261	0.7142	0.6556	0.6998
CODEBERT (MLM, INIT=R)	0.6898	0.6997	0.8383	0.8647	0.7476	0.6893	0.7549
CODEBERT (RTD, INIT=R)	0.6414	0.6512	0.8285	0.8263	0.7150	0.6774	0.7233
CODEBERT (MLM+RTD, INIT=R)	**0.6926**	**0.7059**	**0.8400**	**0.8685**	**0.7484**	**0.7062**	**0.7603**

图3 原论文结果

Model	Ruby	Javascript	Go	Python	Java	PHP	Overall
CodeBERT	0.679	0.620	0.672	0.656	0.672	0.626	0.693
GraphCodeBERT	0.712	0.646	0.896	0.690	0.683	0.649	0.706

图4 复现结果

6 总结与展望

本研究成功复现了 CodeBERT 模型以及加入了图信息的GraphCodeBERT模型，并在自然语言代码搜索任务上取得了与原论文相近的实验结果，验证了模型的有效性。然而，在复现过程中也发现了一些不足之处。例如，在处理一些复杂的代码结构和语义关系时，模型的表现较差，无法处理较长的文本；在模型的训练效率方面，仍有优化的空间，特别是在大规模数据和多语言环境下的训练速度需要进一步提升。

在未来的研究中，如何结合更先进的神经网络架构和训练方法去改善模型对长文本处理能力较差的现象；是否可以引入图神经网络来处理代码的语法和语义结构，或者探索更高效的多模态融合策略，以进一步提高模型在自然语言和编程语言处理任务中的性能。此外，怎么去扩展模型的应用领域，如代码漏洞检测、代码自动修复等，为开发者进行软件开发和维护时，提供强大的工具支持。

参考文献

[1] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

[2] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openaiassets/researchcovers/languageunsupervised/language understanding paper.pdf.

[3] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[4] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237.

[5] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D.,... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

[6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.,... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998 - 6008).

[7] Lu, J., Batra, D., Parikh, D., & Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In Advances in Neural Information Processing Systems, pages 13 - 23.

[8] Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). Videobert: A joint model for video and language representation learning. arXiv preprint arXiv:1904.01766.