University of St Andrews

RepoSim

利用预训练的语言模型进行语义库比较

由
李子豪

督导员：罗莎-费尔盖拉-维森特博士

预科生编号：200007779 圣安德鲁斯

大学

摘要

近年来，自然语言处理（NLP）技术的快速发展刺激了它们在包括软件工程在内的广泛领域的应用。其中一个应用是对源代码进行语义相似性分析，这可以促进代码克隆检测和资源库比较等任务。本论文研究了最先进的预训练语言模型GraphCodeBERT和UniXCoder在执行成对的克隆检测任务和比较资源库的语义相似性时的有效性。

Wefine-tuned GraphCodeBERT和UniXCoder在Python代码对的数据集上进行了克隆检测任务，并评估了它们在C4数据集上的性能。此外，我们还开发了RepoSim，一种有效比较按相关主题分类的资源库的语义相似性的方法，并考察了各种语言模型在这项任务上的表现。我们的评估结果表明，GraphCodeBERT在Python克隆检测任务上取得了最好的性能，而UniXCoderfine在代码搜索任务上的调整，在识别基于其代码嵌入的语义相似的存储库方面表现出色。

本论文对预训练语言模型处理代码相关任务的能力提供了有价值的见解，并有助于理解如何将这些技术应用于捕获资源库的语义信息。这项工作不仅推动了这一领域的研究，也为未来探索前沿的语言模型，如ChatGPT、LLaMa和Bard，在资源库相似性比较和其他相关任务方面铺平了道路。

声明

我声明所提交的评估材料是我自己的作品，除非通过引文或致谢明确归功于他人。该工作是在本学年进行的，除非另有说明。

本项目报告正文长7261字，包括项目规范和计划。

在向圣安德鲁斯大学提交本项目报告时，我允许按照大学图书馆的规定提供报告供使用。我还允许公布报告的标题和摘要，并允许制作报告的副本，以成本价提供给任何真正的图书馆或研究人员，并允许在万维网上提供。我保留这项工作的版权。

鸣谢

我想对我的导师Rosa Filgueira博士表示感谢，她为我提供了机会，让我通过这个项目深入研究机器学习的复杂领域。她的不断指导和支持在帮助我驾驭这项研究的复杂性方面是非常宝贵的。我还想对我的家人和朋友表示感谢，感谢他们在这一年中在经济上和情感上的坚定支持。你们的鼓励对于帮助我实现学术目标至关重要。

内容

图表列表

表格列表

第一章

简介

近年来，软件开发有了很大的进步，代码库的规模和复杂性都在增长。因此，了解不同代码库之间的语义相似性已经成为开发人员、研究人员和组织的一项重要任务。这不仅有助于代码的重用，也有助于识别潜在的代码克隆，从而提高软件维护的效率和软件质量。机器学习（ML）和自然语言处理（NLP）技术，如循环神经网络和变形器，已经成为应对这些挑战的强大工具，并被用于各种代码相关的任务，如代码完成、代码总结和代码搜索。

在这篇论文中，我们探索了使用ML技术进行代码克隆检测和存储库相似性比较。我们对两个预先训练好的模型GraphCodeBERT和UniXCoder进行了一对一的克隆检测任务，并评估了它们的性能。此外，我们开发了一种方法来有效地比较具有不同主题的资源库的语义相似性，并测试了各种语言模型在这个任务中的嵌入性能。我们的评估表明，GraphCodeBERT模型在Python克隆检测任务中表现出色，而UniXCoder模型在代码搜索方面的自我调整在识别基于代码嵌入的语义相似的存储库方面表现最佳。

本论文的组织结构如下：第二章列出了为本项目设定的目标和

的完成状态。第三章提供了关于代码克隆检测的ML和NLP技术的全面背景。第四章介绍了相关的工作和用于比较资源库相似性的工具。第5章深入探讨了模型的实现和实验设置。第6章介绍了评估结果，强调了模型在克隆检测任务和资源库相似性比较上的表现。第7章集中讨论了我们工作的可重现性，详细介绍了对模型进行精细调整和重现评估结果的步骤。最后，第八章是论文的结论，总结了研究结果并概述了未来的研究方向。

我们的工作旨在促进对先进的ML和NLP技术如何应用于代码相关任务的理解，以及它们在捕捉源代码中嵌入的语义信息方面的有效性。通过探索GraphCode- BERT和UniXCoder在克隆检测和资源库相似性比较中的能力，我们希望能阐明这些模型在各种代码相关任务中的潜力，并激发这一领域的进一步研究。

第二章

目标

2.1 主要 目标

1. 对代码间相似性的各种机器学习方法进行全面调查：

   - 基于抽象句法树的模型
   - 基于Python代码嵌入的基于标记的模型：

     -LSTM

     -预先训练过的变形金刚

     -元素的序列
   - 基于元数据和文档串的相似性
   - 基于调用图、数据流、控制流图和存储库结构
   - 基于预先训练好的转化器模型来理解代码

   在这项调查中，我们将讨论这些技术如何工作，并比较它们的性能。

2. 选择一种代码与代码之间的相似性技术，并开发Python脚本，在比较Python代码时评估其性能。

3. 使用公共Python代码数据集对所选模型进行微调，以研究其性能的潜在改进。
4. Enhanceinspect4py可以从一个给定的资源库中获取额外的元数据信息，如Readme、Description、Topics、Tags、Forks、Stars等。在这个过程中，不会提取识别用户的个人信息，我们将只使用具有开放源代码许可证的仓库。
5. 验证从选定的公共资源库中提取的元数据的正确性，确保元数据与源代码的目的相符。我们将对部分资源库进行人工注释，并与inspect4py提取的元数据进行比较。我们的目标是使用超过400个开放源码库来验证所提取的元数据。

2.2 二级 目标

1. 训练或微调一个用于评估存储库之间元数据相似性的模型。用于训练的数据集将来自主要目标中开发的扩展的 inspect4pyd所提取的元数据。
2. 利用这个模型来评估基于元数据的开放源码库的相似性。

2.3 三级 目标

1. 训练或微调一个模型，用于评估基于结构的资源库之间的相似性。每个资源库的结构也将被inspect4py提取出来。
2. 使用这个模型来评估基于其结构的开源存储库的相似性。同样，我们计划使用Repo2Vec论文中指定的相同的资源库集合[1] ，以评估我们模型的性能。

[1]Md Omar Faruk Rokon等人.*Repo2Vec：一种用于判断的综合嵌入方法*

3. 训练一个模型，根据每个资源库的元数据、结构和代码的相似性来评估相似性，并将其性能与之前的模型进行比较。

2.4 目标完成情况 状态

在本论文中，我们已经涵盖了主要目标和次要目标的所有方面，除了作为单独文件提交的背景调查，以及对从资源库中提取的元数据的正确性的验证。元数据的验证没有实现，因为我们使用的是GitHub的官方API来进行元数据提取，所以几乎没有必要验证内容的正确性。

对于三级目标，我们决定放弃使用资源库的结构，而只关注Python资源库的元数据和源代码。后来的评估表明，我们在比较基于这两个部分的资源库的相似性方面取得了很高的性能。

Repository Similarity.2021.doi:10.48550/ARXIV.2107.05112.url:https://arxiv.org/abs/2107.05112.

第三章

基金会

检测和分析代码的相似性在软件工程领域变得越来越重要，因为它可以对代码质量、维护性和可靠性产生重大影响。近年来，各种代码相似性工具已经被开发出来，利用不同的代码表示方法，如抽象语法树（AST），控制流图（CFG），以及深度学习技术，如循环神经网络（RNN）和变形器。

在本章中，我们将深入了解用于确定代码相似性的不同技术。我们将首先定义克隆的类型并介绍常见的代码表示法。接下来，我们将介绍广泛使用的神经网络模型，用于处理各种类型的序列，如句子、段落，甚至代码片段。我们还将探讨这些模型随时间的演变，从最初的用于自然语言处理任务的递归神经网络的发展，到最近的语言模型架构的进步，如BERT、RoBERTa和CodeBERT。最后，我们将介绍本项目中使用的最先进的模型，并探索优化其性能的技术。

3.1 克隆的类型

正如在一项调查中提出的[1]，我们可以将代码克隆的类型分为四类：

- 第一类：相同的代码片段，除了空白处的变化（也可能是布局上的变化）和注释。
- 第二类：结构/句法上完全相同的片段，只是在标识符、字词、类型、布局和注释上有变化。
- 第三类：复制的片段，并作进一步修改。除了标识符、字词、类型、布局和注释的变化外，还可以改变、添加或删除语句。
- 类型四：两个或更多的代码片段，执行相同的计算，但通过不同的语法变体实现。

在这个项目中，我们的目标是检测第四类克隆，即在语义上相似但在语法上可能不同的代码对。

3.2 抽象语法 树

抽象语法树（Abstract Syntax Tree），简称AST，是用正式语言编写的文本（通常是源代码）的抽象语法结构的树状表示。树上的每个节点表示文本中出现的一个结构[2]。它包含了程序的语义和句法信息，因此通常被用来分析源代码之间的相似性。

[1]Chanchal Roy和James Cordy."关于软件克隆检测研究的调查"。In:*School of Computing TR 2007-541*(Jan. 2007).

[2]Wikipedia contributors.*Abstract syntax tree - Wikipedia, The Free Encyclopedia*.https:// en. wikipedia. org/ w/ index. php? title=Abstract_syntax_tree& oldid= 1103626323.[Online; accessed 26-October-2022].2022.

3.3 控制流 图

控制流图或CFG是一种表示方法，使用图的符号，表示程序执行过程中可能穿越的所有路径[3]。它捕获了更全面的语义信息，如代码中的分支、循环和调用关系（由调用图表示），但与AST相比，语法信息要粗略得多[4]。它可以与AST相结合，更好地表示源代码。

3.4 嵌入

在机器学习的背景下，嵌入是在低维向量空间中表示高维数据的过程，如单词、图像或用户偏好。这种技术通常用于自然语言处理任务，其中单词或短语被表示为矢量。通过使用像Word2Vec或BERT这样的神经网络模型从大量的文本数据中学习，我们可以将文本编码为向量，使相似的文本在向量空间中彼此接近，而不相似的文本则相距甚远。

编程语言与自然语言有许多相似之处，因此，自然语言处理中使用的一些技术，包括嵌入，也可以应用于编程语言，以衡量代码的相似性。

3.5 余弦 相似性

余弦相似性是衡量多分类空间中两个非零向量之间的相似性。它测量两个向量之间的角度的余弦，范围从1到1，其中1表示两个向量是相同的，0表示它们是正交的（垂直的），而1表示它们是完全相反的。

[3]Wikipedia contributors.*Control-flow graph - Wikipedia, The Free Encyclopedia*.https : / / en . wikipedia . org / w / index . php ? title = Control - flow _ graph & oldid = 1115609094.[Online; accessed 25-October-2022].2022.

[4]Chunrong Fang等 "Functional Code Clone Detection with Syntax and Semantics Fusion Learning".In:*Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*.ISSTA 2020.Virtual Event, USA: Association for Computing Machinery, 2020, pp. 516-527.isbn: 9781450380089.doi:10.1145/3395363.3397362.url：https://doi.org/10.1145/3395363.3397362.

*双维*向量A和B之间的余弦相似度公式为：[5] ：

$$\cos(\checkmark) = \frac{A\text{-}B}{||A||||B||}$$

余弦相似性可以作为衡量语义相似性的一个指标，它根据词或句子在文本语料库中的出现或共同出现的统计数字，将其含义编码为高维空间中的矢量，矢量空间中的每个维度可以代表含义的一个特征或方面。

3.6 循环神经 网络

递归神经网络或简称RNN是一种旨在处理连续数据的神经网络类型。与大多数传统的前馈神经网络不同，前馈神经网络是逐一处理输入并产生输出的，而RNN有一个递归结构，节点之间的连接可以形成一个循环，使一些节点的输出能够ect随后输入到相同的节点[6] 。这有助于网络保持一个内部状态，或者说记忆，捕捉到它到目前为止所看到的序列信息，这使得它能够更有效地处理序列数据或时间序列数据。

在变换器出现之前，RNN，特别是长短时记忆（LSTM）网络是自然语言处理（NLP）任务的主要工作对象。然而，RNN有几个局限性，包括难以捕捉长期依赖关系，梯度消失问题，以及计算的顺序性导致训练和推理时间缓慢。由于RNN在训练过程中依赖以前的输出来计算下一个输出，它们不能利用并行计算的优势，这就减慢了训练过程。

[5]Wikipedia contributors.*Cosine similarity - Wikipedia, The Free Encyclopedia*.https ：/ / en . wikipedia. org/ w/ index. php? title=余弦相似度& oldid= 1141784488.[Online; accessed 25- March-2023].2023.

[6]Wikipedia contributors.*Recurrent neural network - Wikipedia, The Free Encyclopedia*.https：//en.wikipedia.org/w/index.php?title=Recurrent_neural_network&oldid=1117752117.[Online; accessed 28-October-2022].2022.

3.7 变形金刚

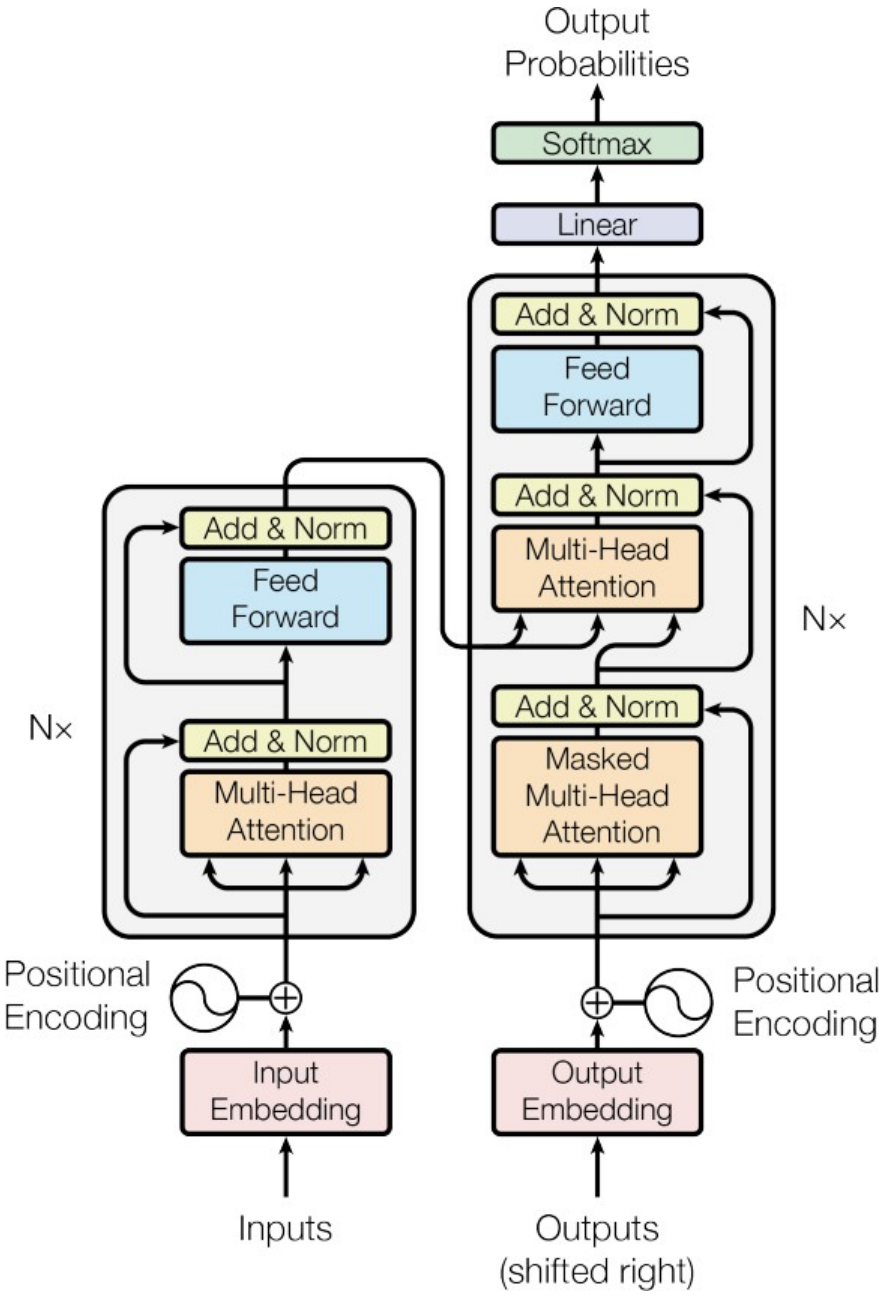

图3.1：变压器结构[7]

[7][8]

变换器是一种深度学习模型，它利用自我注意机制对输入数据的每一部分的重要性进行di↵erentially加权[8]。它也被设计用来处理序列数据，但与逐字处理输入序列的RNN不同，它可以并行处理输入数据，这利用了现代GPU的并行计算能力，使训练过程更加高效。它有两个基本机制：

- 位置编码这涉及到将代表序列中每个元素（例如，词，标记）的位置形成的向量添加到其相应的嵌入中，因此，结果嵌入是有顺序意识的。通过使用位置编码，转化器可以并行地处理序列而不丢失关于序列顺序的信息。
- 自我关注这使得转化器能够有效地模拟序列中元素之间的关系，根据每个元素与同一序列中其他元素的相关性，动态地加权计算其重要性。这也大大降低了每层的计算复杂性，并增加了可并行化的计算量[9]。

现在，转化器越来越成为NLP问题的首选模型，预先训练好的转化器模型，如BERT和GPT，在推动自然语言处理领域的发展中发挥了重要作用。

3.8 BERT

变换器的双向编码器表示法，简称BERT，是一种基于变换器的机器学习技术，用于自然语言处理（NLP）的预训练--。

[8]Wikipedia contributors.*Transformer (machine learning model) - Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/ index.php?title=Transformer_(machine_learning_model)&oldid=1118251522.[Online; accessed 28-October-2022].2022.

[9]Ashish Vaswani等人.*Attention Is All You Need*.2017.doi:10.48550/ARXIV.1706.03762.url: https://arxiv.org/abs/1706.03762。

由谷歌开发的[10]。它有两个预训练目标：掩蔽语言模型和下句预测。掩蔽语言模型随机掩蔽输入序列中的一些词，模型被训练来预测掩蔽的词。下一句预测的任务是预测第二句是否是第一句的延续。这两个任务都可以以无监督的方式进行训练，这使得BERT可以在大型数据集上进行预训练，如BooksCorpus和英语维基百科[11]。预训练的BERT模型能够对语言有一定程度的理解，并可以针对具体的下游任务进行调整。

3.9 罗贝尔特a

RoBERTa（Robustly Optimized BERT pre-training Approach）[12]，是Facebook在2019年开发的BERT的一个修改版本。与BERT相比，RoBERTa使用更大的语料库和更长的时间进行训练，这导致了在各种NLP任务上的性能提高。

3.10 代码BERT

CodeBERT[13]是一个预训练的编程语言模型，使用与RoBERTa相同的架构。它是一个多编程语言模型，在六种编程语言的NL-PL对上进行预训练，包括Python, Java, JavaScript, PHP, Ruby和Go。在预训练阶段，CodeBERT在双模和单模数据上进行训练，其中双模数据是指自然语言-代码对，单模数据仅指自然语言或代码。CodeBERT有两个目标，屏蔽语言建模

[10]Wikipedia contributors.*BERT (language model) - Wikipedia, The Free Encyclopedia*.https://en.wikipedia.org/w/index.php?title=BERT_(language_model)&oldid=1107671243.[Online; accessed 31-October-2022].2022.

[11]Jacob Devlin et al.*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.2018.doi:10.48550/ARXIV.1810.04805.url:https://arxiv.org/abs/1810.04805.

[12]Yinhan Liu等人，*RoBERTa：一种稳健优化的BERT预训练方法*。2019.DOI:10.48550/ARXIV.1907.11692.url:https://arxiv.org/abs/1907.11692.

[13]Zhangyin Feng等人.*CodeBERT: A Pre-Trained Model for Programming and Natural Languages*.2020.doi:10.48550/ARXIV.2002.08155.url:https://arxiv.org/abs/2002.08155.

(如上一节所述）和替换标记检测。在替换代币检测目标中，一些输入代币被替换为由生成器生成的合理的替代物，目标是训练一个模型，以确定损坏的输入中的每个代币是否被生成器样本所替换[14] 。经过预训练和精细调整，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。

3.11 GraphCodeBERT

GraphCodeBERT[15] 是一个由微软亚洲研究院开发的用于源代码处理的预训练语言模型。它是原始CodeBERT模型的一个扩展。

GraphCodeBERT结合了一个图神经网络（GNN）架构来编码源代码中的语法和语义信息，这使得它能够更好地捕捉代码元素之间的结构关系。具体来说，GraphCodeBERT利用抽象语法树（ASTs）来表示代码，并在ASTs上建立一个图结构。图的节点代表代码标记（例如，变量，函数，操作符），边代表这些标记之间的语法关系（例如，AST中的父子关系）。

GraphCodeBERT在大规模的源代码语料库上进行了预训练，并且可以在各种下游任务上进行调整，包括代码分类、代码完成和代码检索。GraphCodeBERT的预训练任务是前面所描述的屏蔽语言建模（MLM）任务和边缘预测任务。在边缘预测任务中，模型通过训练预测数据流中被屏蔽的变量之间的边缘，从数据流中学习表示。

在其发表时，GraphCodeBERT在一些代码相关的下游任务上取得了最先进的表现，包括代码搜索，克隆检测、

[14]Kevin Clark等人.*ELECTRA：预先训练文本编码器作为鉴别器而不是生成器*。2020.doi:10.48550/ARXIV.2003.10555.url:https://arxiv.org/abs/2003.10555.

[15]Daya Guo et al.*GraphCodeBERT: Pre-training Code Representations with Data Flow*.2020.doi：10.48550/ARXIV.2009.08366.url:https://arxiv.org/abs/2009.08366.

代码翻译和代码细化。在这个项目中，一个修改过的GraphCodeBERT模型被微调并作为克隆检测工具使用。

3.12 R-Drop：神经 网络的规则化剔除

被称为R-Drop的正则化策略[16] ，是Dropout技术的一个扩展。它已被证明在应用于精细调整大规模预训练模型（如BERT和RoBERTa）时产生了实质性的改进。

在这个项目中，我们利用了一个基于GraphCode-BERT的定制克隆检测模型[17] ，该模型通过R-Drop技术进行了优化。该模型最初是由thesnoop2headteam开发的，并在Python代码相似性竞赛中表现出卓越的性能，在排行榜上排名前三[18] 。

3.13 对比性的 学习

对比学习是一种机器学习技术，它通过在潜在空间中对正面例子（相似的数据点）和负面例子（不相似的数据点）进行对比来提高数据表示的质量。通过将语义相近的邻居拉到一起，将非邻居推开，对比学习使语言模型能够学习更好的语义表征[19] 。

这种技术也被有效地用于编程语言模型，如C4[20] , SYNCOBERT[21] 和UniXCoder[22] ，以提取代码语义到嵌入。这些

[16]梁晓波等人：*R-Drop：R-Drop: Regularized Dropout for Neural Networks*.2021.doi:10.48550/ARXIV.2106.14448.url:https://arxiv.org/abs/2106.14448.

[17]Park SangHa.*sangHa0411/CloneDetection*.https:// github. com/ sangHa0411 / CloneDetection.2022.

[18]*每月一次的达康码相似度判断人工智能竞赛*。2022年5月.url:https ：/ / Dacon . io / competitions/official/235900/leaderboard.

[19]Tianyu Gao, Xingcheng Yao, and Danqi Chen.*SimCSE: Simple Contrastive Learning of Sentence Embeddings*.2021.doi:10.48550/ARXIV.2104.08821.url:https://arxiv.org/abs/2104.08821.

[20]Chenning Tao等 "C4: Contrastive Cross-Language Code Clone Detection".In:*2022 IEEE/ACM第30届国际程序理解会议（ICPC ）*。2022, pp. 413-424.doi:10.1145/ 3524610.3527911.

[21]王欣等.*SynCoBERT: Syntax-Guided Multi-Modal Contrastive Pre-Training for Code Repre- sentation*. 2021.doi:10.48550/ARXIV.2108.04556.url:https://arxiv.org/abs/2108.04556.

[22]Daya Guo et al.*UniXcoder：UniXcoder: Unified Cross-Modal Pre-training for Code Representation*.2022.doi：

在涉及各种编程语言和自然语言之间的语义相似性比较的任务中，如代码搜索和克隆检测，模型有很强的性能，正如他们的评价所表明的那样。

3.14 编码机（UniXCoder

UnixCoder是另一个基于转化器的预训练模型，它利用包括AST和代码注释在内的跨模式内容来增强代码表示[23]。

为了通过嵌入更好地表示代码片段的语义，作者引入了两个预训练任务：多模态对比学习（MCL），利用AST获得更全面的代码语义表示；跨模态生成（CMG），利用代码注释来调整编程语言间的嵌入。

UniXcoder在克隆检测和代码搜索任务中超过了以前最先进的模型，包括CodeBERT、Graph- CodeBERT、SYNCOBERT和CodeT5，这在论文的实验中得到了证明。作者还提出了一个新的任务，称为零点代码到代码搜索，以评估代码片段嵌入的性能。在这个任务中，一个源代码被用作查询，以便在零次拍摄的情况下从候选代码集合中检索具有相同语义的代码。UniXcoder达到了最先进的性能，并且超过了GraphCodeBERT，总分提高了约11分。本文的消减研究表明，对比学习和多模态的使用对该模型的改进贡献最大。

鉴于UniXCoder在提取代码语义信息方面的强大性能及其在各种编程语言中的适用性，我们将在本项目中采用它来提取代码嵌入并在它们之间进行余弦相似度比较。

10.48550/ARXIV.2203.03850.url:https://arxiv.org/abs/2203.03850.

[23]Daya Guo et al.*UniXcoder：UniXcoder: Unified Cross-Modal Pre-training for Code Representation*.2022.doi:

10.48550/ARXIV.2203.03850.url:https://arxiv.org/abs/2203.03850.

第四章

相关作品/工具

4.1 inspect4py

Inspect4py[1] 是一个静态代码分析框架，旨在帮助开发人员理解和分类Python软件库，通过分析和提取重要信息，如函数、类、方法、文档、依赖关系、调用图和控制流图。它对于理解、管理和维护软件库非常有用。

该项目包括扩展inspect4py的功能，使其能够从给定的资源库中提取额外的元数据信息，如Readme、描述、主题、标签、叉子、星星等，这有助于识别具有类似特征的资源库，如作者、主题、描述等等。

该工具提取函数源代码和文档字符串的能力被用来将输入库转换成函数源代码和文档字符串的列表。然后，深度学习模型可以用来计算这些列表中每个代码或文档字符串的嵌入。

[1]Rosa Filgueira 和 Daniel Garijo."Inspect4py：Python代码库的知识提取框架"。In:*2022 IEEE/ACM第19届国际软件资源库挖掘会议（MSR）*。2022, pp. 232-236.doi:10.1145/3524842.3528497.

4.2 呼叫中心

Repo2Vec[2] 是一种被提出来的方法，使用来自三种类型的信息：元数据、目录结构和源代码，在一个嵌入矢量中表示GitHub仓库。该方法遵循一个由四个步骤组成的管道：

1. meta2vec元数据嵌入是通过使用doc2vecm模型将资源库的元数据映射到一个嵌入向量中来计算的。元数据包括诸如资源库名称、描述和语言等信息。

2. struct2vecDirectory结构嵌入是通过以下三个步骤计算的：首先，目录结构被转化为树状表示。其次，使用thenode2vecmodel生成每个节点的向量。最后，节点向量被合成为一个单一的结构向量，使用列式聚合方法来表示目录结构。

3. source2vec源代码嵌入也是通过以下三个步骤计算的：首先，使用code2vecm模型生成每个方法的向量。其次，汇总每个文件中的所有方法代码向量，得到文件向量。最后，将所有文件向量汇总，得到整体代码向量。

4. repo2vec最后，存储库的嵌入是通过串联前面步骤得到的嵌入来计算的。每种嵌入类型的权重由用户决定，可以调整为对某些信息源给予更多或更少的重要性。

Repo2Vec的评估显示了利用多个来源的信息来有效地表示一个资源库的有效性，它也强调了包括元数据嵌入对整体性能的重要性。根据这种方法，我们的项目旨在比较两个资源库时，将代码嵌入的相似性和文档嵌入的相似性都包括在内。通过考虑多种来源的

[2]Md Omar Faruk Rokon 等人. *Repo2Vec：一种确定存储库相似性的综合嵌入方法*。2021.doi:10.48550/ARXIV.2107.05112.url:https://arxiv.org/abs/2107.05112.

信息，我们可以更全面地了解一个存储库的内容，并最终提高我们准确比较类似存储库的能力。

第五章

实施

在这一节中，我们介绍了我们比较Python源代码和资源库之间语义相似性的实现过程。首先，我们讨论了两种方法：一种是使用改进的GraphCodeBERT模型的交叉编码器技术，另一种是使用UniXCoder模型的双编码器方法。接下来，我们概述了使用特定数据集对这些模型进行微调的过程，并详细分析了它们的优势和劣势。随后，我们描述了我们比较资源库相似度的方法，强调了双编码器方法在效率方面的优势。最后，我们解释了根据我们对预训练的句子转化器模型的评估，选择表现最好的文档串嵌入模型。

5.1 代码相似性 比较

在这个项目中，我们利用两个预先训练好的语言模型来比较Python源代码之间的语义相似度。具体来说，我们采用了一个经过修改的Graph- CodeBERT模型，该模型在克隆检测任务中进行了微调，以及一个UniXCoder模型，该模型在自然语言代码搜索任务中进行了微调。这些模型使用两种不同的范式产生相似性结果：交叉编码器和双编码器，这两种范式将在下面的小节中详细介绍。

5.1.1 交叉编码器和双 编码器

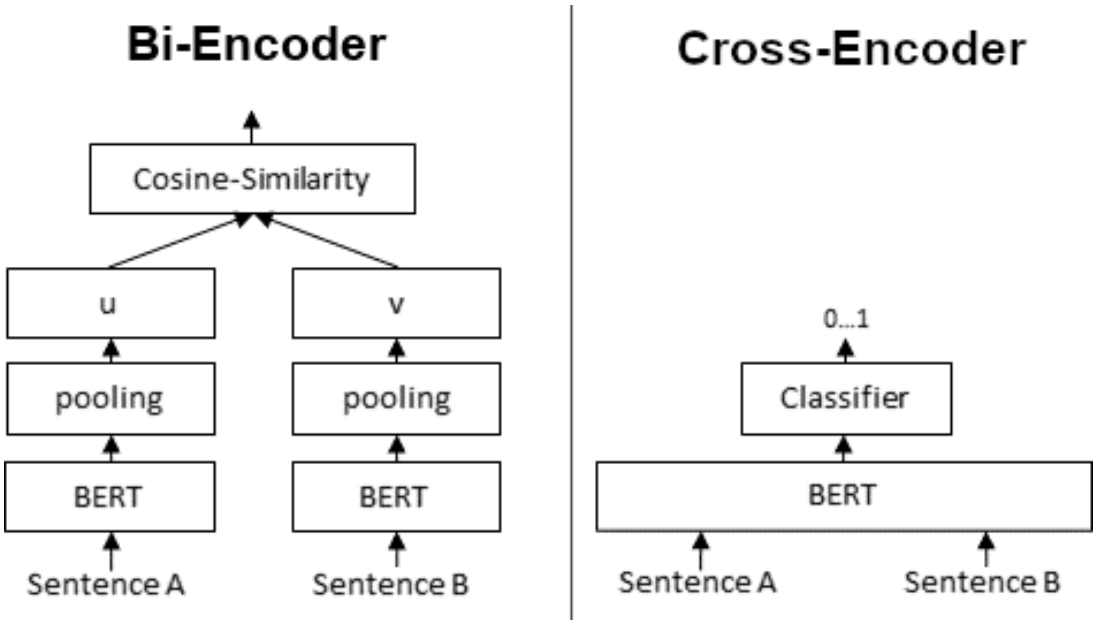

图5.1：双编码器与交叉编码器的对比[1]

在比较句子的语义相似性时，有两种类型的架构被普遍使用：双编码器和交叉编码器。每种架构都有自己的优势和劣势。

双编码器对每个句子进行独立编码，并产生句子嵌入，随后可以使用余弦相似度等相似度函数进行比较。由于它们只需要对每个句子进行一次编码，因此非常适合于涉及大量句子比较的任务，如聚类和信息检索。

另一方面，交叉编码器接收成对的句子，并在编码前将它们连接起来。在克隆检测任务中，使用线性分类器对编码器的输出嵌入进行分类，以确定该句子对是否是克隆的。这种架构在语义相似性比较方面通常比双编码器表现得更好，因为它可以在编码过程中捕获两个输入句子的全局背景，而双编码器只考虑局部背景，因为它们在编码每个

[1][24]

句子分开。

5.1.2 方法1：GraphCodeBERT作为一个跨 编码器

与Java或C不同，它们已经有了现有的克隆检测基准（有确认的克隆对的数据集），如BigCloneBench[2] 和POJ-104[3] ，Python代码的第四类克隆数据集则很少。我们最终设法找到了两个包含标记代码对的数据集：

1. 5倍的数据集：我们最初发现了一个由韩国人工智能黑客马拉松平台Dacon[4] 举办的代码相似性竞赛。这个比赛的目的是寻找在检测语义相似的Python代码方面表现最好的模型。Thesnoop2head团队在这次比赛中排名前三，最终得分是0.98061。为了重现性，他们将他们的模型[5] 到GitHub，并将微调数据集[6] 到Hugging Face Hub。该数据集在训练集中包含5,388,622个标记的代码对，在测试集中包含1,324,360个。

2. C4数据集：我们还发现了另一个较小的数据集，它包含多种编程语言的语义相似的克隆对。这个数据集最初用于研究论文C4：Contrastive Cross-Language Code Clone De- tection[7] ，用于检测不同编程语言之间的克隆。我们对该数据集进行了过滤，使其只包含Python克隆对，然后生成了负的

---

[2] Je↵rey Svajlenko等人，"Towards a Big Data Curated Benchmark of Inter-Project Code Clones"。In：*2014年IEEE国际软件维护与演进会议论文集*。ICSME '14。USA: IEEE Computer Society, 2014, pp. 476-480.isbn: 9781479961467.doi:10.1109/ICSME.2014.77.url:https://doi.org/10.1109/ICSME.2014.77.

[3] Lili Mou等人 "Convolutional neural networks over tree structures for programming language processing" 。In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.2016年，第1287-1293页。

[4] *每月一次的达康码相似度判断人工智能竞赛*。2022年5月.url:https ：/ / Dacon . io / competitions/official/235900/leaderboard.

[5] Park SangHa.*sangHa0411/CloneDetection*.https:// github. com/ sangHa0411 / CloneDetection. 2022.

[6] Young Jin Ahn.*Clone-Detection-600k-5fold*. Hugging Face Datasets. 联系邮箱：youngahn@yonsei.ac.kr. 2022.url:https ：/ / huggingface . co / datasets / PoolC / 1 - fold - clone - detection-600k-5fold.

[7] Chenning Tao等 "C4: Contrastive Cross-Language Code Clone Detection".In:*2022 IEEE/ACM第30届国际程序理解会议（ICPC ）*。2022, pp. 413-424.doi:10.1145/ 3524610.3527911.

通过随机选择每个阳性克隆对中第一个代码的非克隆代码，对样本进行分析。这就产生了一个平衡的数据集，其中有6,852个阳性样本和同等数量的阴性样本。

在5折数据集上的微调我们利用修改后的GraphCodeBERT模型和由thesnoop2headteam上传的数据集，试图重现他们的工作。由于时间和资源有限，我们只在5折数据集的第一折上使用A40 GPU服务器进行了约11.5小时的模型微调。然后我们选择了评价分数最高的检查点进行后续实验。

5.1.3 方法2：UniXCoder作为一个双 编码器

尽管GraphCodeBERT模型在后来的克隆检测评估中表现出了有效性，但我们发现使用它来比较具有众多功能的大型存储库是不切实际的。

为了解决这个问题，我们选择了一个双编码器的方法。虽然我们微调的GraphCodeBERT模型善于确定一个输入句子对是否是克隆的，但当它作为一个双编码器使用时，即分别对每个代码进行编码并比较其输出嵌入的余弦相似度时，其性能就不那么强。因此，UniX- Coder通过对比学习生成包含语义信息的更好的嵌入的能力使它成为比以前的GraphCodeBERT模型更可取的选择。

UniXCoder - 对克隆检测任务进行了微调

由于UniXCoder的作者提供的精细调谐脚本不如用于定制GraphCodeBERT模型的脚本有效，所以使用了较少数量的代码对进行精细调谐。因此，我们在A40服务器上对来自5fold数据集的大约13,000个平衡的代码对进行了UniXCode的精细调整，时间大约为10小时。

尽管该模型是使用交叉编码器范式进行微调的，但当对C4的语义相似的代码对进行评分时，它仍然表现出良好的性能。