 DOI: https://doi.org/10.12345/bdai.v5i5.18057

Research on Plagiarism Detection Methods for Open-ended Programming Questions in Higher Education Self-study Exams

Rui Li Yixiong Zhang Qiang Guo Yu Wang Yan Bai

School of Continuing Education, Xi'an Jiaotong University, Xi'an, Shaanxi, 710049, China

Abstract

The software program design course practice assessment of higher education self-study examination usually adopts open programming questions, and plagiarism detection is an important task to evaluate students to submit programming homework. This paper proposes a deep learning-based similarity detection model to solve the problem of cross-morphological similarity detection between source code and binary code. The control flow graph information is extracted through Joern and IDA Pro tools, and multimodal features such as sequence token, character font and control flow graph structure are fused, which are processed by a semantic encoding network to generate representative vectors. This paper introduces a contrast-learning model for extracting semantic features of binary code. Triplet loss-guided representation learning is used to adjust the feature representation of the source code with the binary code to achieve code similarity detection across morphologies. Experimental results show that the proposed method maintains a high accuracy of 93.21% in the presence of compilation interference. Future research directions include further analysis of the contributions of each mode feature and exploring the possibility of introducing new modes for similarity detection.

Keywords

practical assessment; programming problems; plagiarism detection; pre-trained mode

高等教育自学考试实践考核中开放式编程题的抄袭检测方法研究

李睿 张轶雄 郭强 王昱 白雁

西安交通大学继续教育学院，中国·陕西 西安 710049

摘 要

高等教育自学考试的软件程序设计课程实践考核通常采用开放式编程题目，抄袭检测是评估学生提交编程作业的重要任务。论文提出了一种基于深度学习的相似性检测模型，以解决源代码与二进制代码之间的跨形态相似性检测问题。通过Joern和IDA Pro工具提取控制流图信息，并融合序列token、字符字面量和控制流图结构等多模态特征，这些特征经过语义编码网络处理生成代表向量。论文介绍了一个对抗编译干扰的对比学习预训练模型，用于提取二进制代码的语义特征。采用三联体损失引导的表示学习，调整源代码与二进制代码的特征表示，实现跨形态的代码相似性检测。实验结果显示，该方法在编译干扰存在情况下能保持93.21%的高准确率。未来研究方向包括进一步分析各模态特征的贡献，并探索引入新模态进行相似性检测的可能性。

关键词

实践考核；编程题；抄袭检测；预训练模型

【课题项目】中国高等教育学会2022年度高等教育科学研究规划课题《支撑S2B2C模式的自考助学智慧服务平台建设》（项目编号：22ZXKS0308）；中国高等教育学会2023年度高等教育科学研究规划课题《面向自考网络助学的群体协同学习研究与探索》（项目编号：23ZXKS0305）；陕西高等教育教学改革研究项目《陕西省高等教育自学考试网络助学平台数字基座建设研究与实践》（项目编号：23JG001）。

【作者简介】李睿（1983–），男，中国陕西汉中人，博士，高级工程师，从事智慧教育与数字化考试研究。

1 引言

高等教育自学考试实践考核评估学生运用专业知识解决实际问题的能力。开放式题目有效评估综合能力、创新思维和实际应用能力，适用于实践考核，特别是软件程序设计课程。这类题目没有固定答案，允许多种解决方案，更注重方法、创新和代码质量。

开放式编程题评估考生的算法、数据结构、软件工程和用户界面设计能力，鼓励创意和独特的解决方案。它们不仅评估个人技能，也用于团队项目和协作能力。在这类考核中，抄袭检测是一个重要且具有挑战性的任务。开放式题目允许多种解决方案，结构和实现上可能差异较大，但也存在

提交他人代码的风险。

有效的抄袭检测机制维护考核公正性和完整性。学生的编程作业可能包括源码和调用的包或库的二进制代码。为了掩饰抄袭行为，学生可能将部分源码编译为二进制代码，增加检测难度。因此，需要同时考虑源代码和二进制代码的抄袭检测。

代码相似性分析是检测抄袭的核心技术，备受工业界和学术界关注。相似性分析揭示作业间的结构、逻辑和文本相似度，判断潜在抄袭行为。当前技术包括文本比较、语法树比较、标记序列比较和代码“指纹”比较，但在同时检测源码和二进制代码时，传统方法会失效。

针对这种跨形态代码相似性检测的研究迫在眉睫。论文提出一种方案，使用 CodeBERT 预训练模型提取源代码语义表示向量，采用对比学习的预训练语义编码网络处理二进制代码，进行多模态特征融合，解决跨形态代码相似性比较的复杂性。论文第二章介绍相关工作，第三章探讨预处理方法，第四章探讨语义编码方法，第五章探讨特征融合和相似性检测，第六章总结与展望。

2 相关研究

2.1 特定形态下的代码相似性检测方法

2.1.1 面向源代码的代码相似性检测方法

代码相似性检测技术发展迅速，已有较为成熟的项目。例如，Moss（Measure of Software Similarity）广泛应用于教育领域，支持多种编程语言，用户提交代码后可接收相似性报告，用于预防和检测作业抄袭。

跨语言的源代码相似性检测技术也在不断成熟，例如 JPlag，专为教育领域设计，用于发现和防止学生作业抄袭。JPlag 支持 Java、C++ 和 Python 等多种语言，能够分析这些语言之间的代码相似性，提供网页界面展示匹配结果，适用于不同编程语言项目的抄袭检测需求。

2.1.2 面向二进制代码的代码相似性检测方法

二进制代码相似性研究起步较晚。Baker 及其团队在 20 世纪末提出了一种压缩和比较可执行代码差异的方法，并开发了 EXEDIFF 工具。随着领域的发展，形成了六种主要检测技术，包括基于属性度量、代码语法、代码结构、代码语义、行为分析和代码表示学习的方法。例如，α Diff 工具通过字节流信息构建特征向量并判断相似性；Asm2Vec 采用自监督学习框架，将汇编指令序列转换为嵌入向量，通过余弦距离评估相似性。

2.2 跨形态的代码相似性检测方法

二进制代码由源代码经编译器编译而成。由于编译过程中丢失了变量名和树结构信息，取而代之的是指令、编译器优化和寄存器信息，源代码和二进制代码在字面表示上差异巨大。跨形态代码相似性检测需要解决这些差异，复杂性较高，因此相关研究较为罕见。

目前的方法主要依赖于提取代码的字面特征，如字符串、整数字面量和递归结构，通过匹配算法评估相似度。然而，这种方法有局限性：函数代码中的字面量信息有限，忽略深层语义特征会影响准确性；且特征选择和工程需要丰富的专家经验，过程耗时且效率低。2019 年，中国科学院信息工程研究所推出了 B2SFinder 技术，检测商业软件对开源项目的重用。此外，CodeCMR 使用独立的神经网络模型从源代码和二进制代码中学习表示向量，通过三元组损失网络实现跨模态融合，在函数级别实现跨形态相似性匹配。

现有方法主要使用传统字符匹配算法来比对字面量特征，存在效率低和准确度有限的问题。由于源代码和二进制代码存在本质差异，跨形态代码相似性研究面临重重困难。设计一个融合多种模式的跨形态代码相似性检测方案，实现源代码与二进制代码之间的相似性检测任务，尤为重要。

3 代码原始表示的抽取与预处理

对于代码，粗粒度信息包括整数字面量和字符串字面量，可用于表示代码。代码的序列 token、控制流图和抽象语法树信息也可表示代码。代码相似性检测的第一步是分别提取源代码和二进制代码的这些特征信息。

3.1 源代码原始表示抽取及预处理

论文聚焦函数级别的相似性检测。虽然函数级相似性检测应用广泛，但代码量减少使得特征变得稀缺。为了提高模型准确性，需要提取更细致的特征，并采用匹配的表示学习模型。

源代码与二进制代码之间，字符串保持一致，意味着相似代码对包含相同的字符串字面量。尽管字符串信息较少，但可作为补充信息。代码的 token 序列表示源代码与二进制代码中的序列和指令顺序，是细粒度信息。此外，代码结构特性是相似度检测的关键因素，显示代码执行时的调用流程，具有抗混淆能力。

①字符串字面量抽取：遍历源函数，遇到双引号表示发现字符串字面量，将其内容提取并单独存储。存储时，将字符串内部空格替换为“-”，字符串间用空格分隔。若源函数不含字符串信息，则返回空值。

② token 序列信息抽取：将源函数转化为 token 序列，删除与代码语义无关的内容，如注释和空行，然后分词并标准化处理，生成适合模型训练的 token 序列。

③结构信息抽取：代码结构特征蕴含丰富的代码语义信息，能洞察代码执行过程中的调用流。本研究选用控制流图代表代码结构信息，并将其融入相似性检测模型。控制流图展示程序处理过程中的所有可能执行路径。若源代码与对应二进制代码相似，则逻辑顺序和跳转流程也应相似，其控制流图信息也会相似。

为提取源代码的控制流图信息，论文利用轻量级开源工具 Joern，该工具能基于静态源代码块自动解析并生成各类代码属性图，适合函数级别的相似性分析任务。

3.2 二进制代码原始表示抽取及预处理

3.2.1 字符串字面量抽取

使用 IDA Pro 反汇编工具获取二进制代码中的字符串字面量。通过查看“Strings”窗口内的所有字符串信息，提取目标代码中的字符串字面量。

3.2.2 token 序列信息抽取

使用 IDA Pro 工具反汇编二进制代码，将反汇编后的函数代码按行为单位存储。将空格替换为“-”作汇编指令内部的分隔符。将所有整数值替换为“dig”标识符，减少具体数值对相似性分析的影响。将不同通用寄存器名称统一替换为“reg”表示。一条汇编指令视为一个 token，指令的 token 序列通过空格分隔。这一从原始汇编代码到目标代码 token 序列的转换流程，如图 1 所展示。

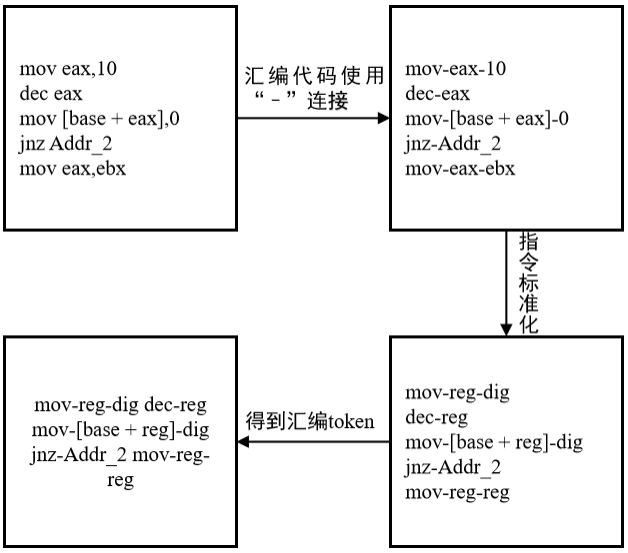


图 1 汇编 token 处理流程

3.2.3 结构信息抽取

IDA Pro 不仅反汇编二进制代码，还能提取二进制函数的控制流图（CFG）信息。通过分析指令的跳转逻辑，函数被划分为基本块，这些基本块构成控制流图的节点，并根据跳转关系连接这些节点形成控制流图的边。通过记录每个节点的基本代码块和节点间的跳转关系，完成对二进制代码控制流图信息的抽取。

在存储控制流图（CFG）信息时，逐行记录每个节点所代表的基本代码块，并使用独立行描述节点间的跳转关系。例如，图 2 中展示的一段汇编代码及其控制流图信息，通过分析指令间的跳转逻辑，识别出函数通过分支结构实现功能。

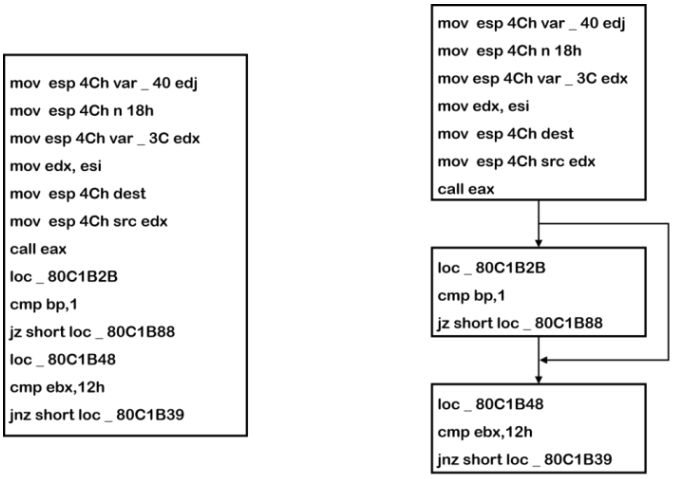
4 代码语义编码网络

源代码与二进制代码差异大，不能直接比较相似性。语义编码网络能将文本转为反映原信息的向量，适用多种任务。代码信息有其语义和语法。本研究为源代码和二进制代码构建语义编码网络，将特征信息转化为包含语义和语法的向量。针对不同模态特征，相似性检测模型需设计特定的语义编码网络，以提取和表达这些模态特征向量。

4.1 基于预训练模型增强的源代码语义编码

BERT（Bidirectional Encoder Representations from Transformers），由 Google 于 2018 年提出，是 NLP 领域的一大创新。它基于 Transformer 双向编码器，通过自监督在大数据上预训练，深入学习文本的上下文关系，掌握复杂语境。BERT 具有出色的迁移学习能力，可通过微调适应具体 NLP 任务，处理复杂语义和长距离依赖问题。

Feng 及其团队首次将 BERT 引入代码处理领域，提出了 CodeBERT 模型。CodeBERT 结合了代码和相关注释的双模态数据进行训练，展现出卓越的泛化能力，适用于多语言代码检测任务，包括 Java、Python、Go 和 C 语言等。论文采用 CodeBERT 模型提取源代码序列的特征。



(a) 二进制代码　　(b) 二进制代码控制流图

图 2 目标代码及其控制流图信息

CodeBERT 遵循 BERT 架构，含 12 个编码层，每层含 12 个自注意力单元（尺寸 64）和全连接层，通过大规模语料训练优化。应用 CodeBERT 于源代码特征表征时，以代码 token 为输入，通过内置词嵌入模块和特征提取部分，将代码序列转为表征性特征向量。

4.2 基于对比学习预训练的二进制代码语义编码

编译器配置差异会影响二进制代码相似性分析。论文提出一种孪生网络和对比学习的二进制语义编码模型，减小相同源代码编译结果的差异，提升模型鲁棒性和抗干扰能力。

4.2.1 二进制样本对构建

基于代码 token 嵌入的相似性构建样本对。具体地，假定两段二进制代码通过 token 嵌入后转为向量 ϑ_a与 ϑ_b，使用余弦距离衡量相似度，公式如下：

$$Dis(Bin_a, Bin_b) = cosine(\vartheta_a,\ \vartheta_b) = \frac{\vartheta_a \cdot \vartheta_b}{\|\vartheta_a\|\|\vartheta_b\|}\#\quad（1）$$

$$Similarity(Bin_a, Bin_b) = 1 - Dis(Bin_a, Bin_b)\#\quad（2）$$

设置合适的相似性阈值对数据代码对构建至关重要。不相似阈值过低或过高、相似阈值过低会影响模型训练和样本对区分，降低准确性。论文中相似性得分＞0.8 表示相似，＜0.3 表示不相似。

确定正负样本对阈值后，构建二进制代码对数据集，每个样本三元组表示为 $(Bin_a, Bin_b, label), label = 1$，表示相似（正样本），$label$ =0 不相似（负样本）。数据集采用 GCC、ICC、Clang 编译器的 O0、O3 优化选项编译的汇编代码，包含 125,692 对正负样本，其中正样本 62,352 对，负样本 63,340 对。

4.2.2 二进制代码编码模型对比预训练

使用 GCC 编译器 O0 优化选项编译的二进制代码与源代码组成的对在 CMSim 模型中相似性检测准确，但换成 GCC 的 O3 优化或 Clang 的 O0 优化编译的二进制时，检测效果下降。CMSim 模型原先只训练 GCC O0 编译的数据集，难以适应不同编译选项导致的代码差异。为减少编译选项影响，引入基于对比学习的二进制代码语义编码网络，通过学习同源二进制码对的共性和非同源对的差异，提高编码效果。

对比学习驱动的二进制语义编码网络模型采用 Text CNN、GCN 和 Bi-LSTM 分别提取 token 序列、结构和字符串字面量特征，如图 3 所示。论文利用孪生网络简化训练，目标是缩小同源二进制代码对距离，扩大非同源对距离。

以源代码通过 GCC 的 O0 和 O3 优化编译得到的二进制代码 g_a与 g_b为例，孪生网络处理这对代码，通过表示学习模型提取特征向量并融合得到最终表示 x_a 和 x_b。模型在提取三种模态特征后，添加网络模块利用对比学习优化编码。定义相似性以保证对称性 $sim(x_a, x_b) = sim(x_b, x_a)$，然后通过向量拼接、多层感知机处理，并通过 sigmoid 函数得到 [0,1] 区间的相似性结果。使用 Cosine loss 计算损失，训练得到的网络可应用于跨形态代码相似性检测。

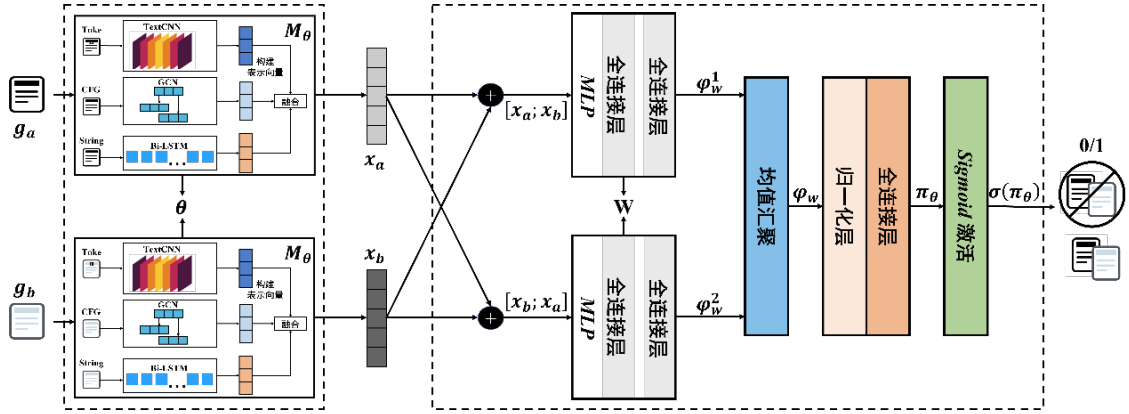


图 3 二进制代码对比学习模型

5 基于多模态特征融合的代码相似性检测

不同代码表示形式可以视为代码的单模态特征信息。多模态特征融合将各种单模态特征映射到共享语义子空间，便于整合。多模态表示构建处理和聚合多种模态信息的模型[1,2]，分为联合表示和协调表示两类。

5.1 基于联合表示策略的形态内表示向量融合

通过不同语义编码网络编码代码的字符字面量、token 序列和控制流图特征，得到三种模态的特征向量。联合表示将这些单模态信息映射到同一多模态空间中。实现联合表示的方式如下：

$$x_m = f(x_1, x_2, x_3, ..., x_n)\#\quad（3）$$

式（3）中，f为融合机制，它可以是个深度神经网络，x_1、x_2 等符号分别是指各模态信息经过表示学习生成的模态特征向量，x_m 是经过联合表示方法完成特征融合的融合向量。

对源代码函数 c_1，提取 token、字符字面量和结构特征，用适配的模型 α_1、α_2、α_3 训练得到各模态特征向量。

$$c_1^{tok} = \alpha_1(c_1)\#\quad（4）$$

$$c_1^{str} = \alpha_2(c_1)\#\quad（5）$$

$$c_1^{cfg} = \alpha_3(c_1)\# \quad (6)$$

其用 c_1^{tok}、c_1^{str}、c_1^{cfg}分别表示源代码 c_1 的不同模态信息经过适配的语义编码网络生成的表示向量，使用联合表示方法融合这三个向量，其公式如下：

$$c_s = f\left(c_1^{tok}, c_1^{str}, c_1^{cfg}\right)\# \quad (7)$$

式（7）中，c_s 代表经过联合表示生成的源代码融合向量；f 为模态特征融合机制。

对于某一段目标代码片段 c_2，针对该函数代码的 token 序列特征，字符特征及结构特征信息设计适配的表示学习模型 B_1、B_2 和 B_3，经过模型的训练可以得到二进制代码各个模态的特征向量。

$$c_2^{tok} = \beta_1(c_2)\# \quad (8)$$

$$c_2^{str} = \beta_2(c_2)\# \quad (9)$$

$$c_2^{cfg} = \beta_3(c_2)\# \quad (10)$$

其用 c_2^{tok}、c_2^{str}、c_2^{cfg}分别表示二进制代码 c_2 的不同模态信息经过适配的语义编码网络生成的表示向量，使用联合表示融合这三个向量，其公式如下：

$$c_b = f\left(c_2^{tok}, c_2^{str}, c_2^{cfg}\right)\# \quad (11)$$

其中 c_b 为经过联合表示后生成的二进制代码融合向量。

特征融合方法多样，主要包括逐点相加和直接拼接两种。论文采用向量直接拼接的方法来完成多模态代码的融合任务。定义代码模态信息的维度分别为 m、n 和 k，则对于代码的三种模态特征信息可以记作：$c^{tok}\epsilon\mathbb{R}^n$、$c^{str}\epsilon\mathbb{R}^m$ 和 $c^{cfg}\epsilon\mathbb{R}^n$，使用向量拼接法实现特征融合得到的向量为 $Con_s = [c^{tok}; c^{str}; c^{cfg}], Con_s \in \mathbb{R}^{n+m+k}$，使用该方法完成表示向量融合任务得到的特征向量 Con_s 的维度为 320。

定义 c_s 与 c_b 分别作为源代码及二进制代码的融合强语义代码表示向量。对于融合机制 f 的设计，考虑到不同模态的代码特征表示是从代码的各个角度对代码进行的互补性描述，对三种信息采用简单的线性融合方式即：

$$c_s = \left[c_1^{tok}; c_1^{str}; c_1^{cfg}\right]\# \quad (12)$$

$$c_b = \left[c_2^{tok}; c_2^{str}; c_2^{cfg}\right]\# \quad (13)$$

5.2 基于协调表示策略的形态间表示向量对齐

协调表示在多模态学习中，通过相似性约束分别处理单模态信息，使其映射到中间语义空间，形成统一的表示形式。由于源代码 c_s 和二进制代码 c_b 形式不同，相似性分析需要一致的表示方式。

协调表示是指在多模态学习或数据融合中，通过在相似性约束下分别处理各单模态信息，引导它们映射到中间语义空间，使其具备统一的、具有相同维度的表示形式。因源函数代码 c_s 和二进制函数 c_b 代码存在不同的形式，所以在进行相似性分析时，需要采用一致的表示方式来处理这两种代码。

每种代码形态有一个特定投影函数 d_1，源代码的投影函数为 d_2，二进制代码的投影函数为。这两个投影函数将代码映射到一个相似性约束的协调中间语义空间中，即：$d_1(c_s)$~$d_2(c_b)$。

协调表示方法包括最小化余弦距离和最大化相关性等技术。论文使用余弦相似性度量法对齐不同形态间的表示向量，以实现基于协调表示的策略。

5.3 基于 Cosine 距离的跨形态代码相似性检测

通过计算两个向量间夹角的余弦值评估相似度，因此也称为余弦相似度或 Cosine 距离。其优点在于，向量间相似度仅取决于夹角，不受向量维度影响，提高了代码相似性检测模型的准确度。余弦值范围为 [-1,1]，夹角越小，余弦值越接近 1，表示相似度越高。

对于相同子空间中的两个向量量 $v_1(x_1, y_1)$ 和$v_2(x_2, y_2)$，其相似度可依据公式（14）和（15）进行计算，其中 θ 表示向量 v_1 和 v_2 之间的夹角。

$$dist(v_1, v_2) = cos\theta \quad (14)$$

$$cos\theta = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|} = \frac{x_1x_2 + y_1y_2}{\sqrt{x_1^2 + y_1^2}\sqrt{x_2^2 + y_2^2}}\# \quad (15)$$

为计算特征向量间的距离，将此公式扩展到两个 m 维向量，即计算 $X(x_1, x_2, x_3, ..., x_m)$ 向量与 $Y(y_1, y_2, y_3, ..., y_m)$ 向量之间的相似度，如公式 3.23 所示。

$$dist(X, Y) = \frac{\sum_{i=1}^{m} x_i y_i}{\sqrt{\sum_i^m x_i^2 \sum_i^m y_i^2}}\# \quad (16)$$

论文使用 $x = \{x_1, x_2, x_3\}$ 来代表融合后的源代码表示向量，$y = \{y_1, y_2, y_3\}$ 代表融合后的二进制代码表示向量，根据公式（16）来计算源代码 x 与目标代码 y 之间的距离。

在相似性检测任务中，需设定样本相似性阈值。检测分数超过阈值则代码相似，反之则不相似。论文默认阈值为 0.8，并评估其对模型的影响。通过计算损失和更新参数，完成跨形态代码相似性检测。

5.4 三联体损失引导的语义编码跨形态协同表示学习

在监督学习中，损失函数用于衡量预测值与真实值的差异。模型性能与损失函数的数值密切相关，性能越好的模型，其损失数值通常越低。

论文采用三联体损失 [3] 的跨形态协同表示学习，通过映射空间度量样本间距离，让相似样本靠近，不同样本远离。三联体损失又被称为 Triplet loss，通过构建包含基础样本、正样本和负样本的数据三元组，确保基础样本与正样本之间的距离小于与负样本之间的距离，并使用距离边界 margin 来进行区分。Triplet loss 计算如 $Loss = Max(Dis(B, P) - Dis(B, N) + margin, 0)$，其结果分为三种情况：Easy Triplets（损失为 0，无需训练）、Hard Triplets（正样本距离大于负样本，损失大于 margin）、Semi-Hard Triplets（正样本距离小于负样本但未满足 margin，损失介于 0 与 margin 之间）。

论文利用含 125,692 对样本的数据集预训练二进制语义

编码网络，减少编译选项干扰。通过三联体损失优化模型，使得相似的源 - 二进制代码对的表示向量接近，而不相似的远离。

对于源代码和其对应的两个二进制代码版本 b_1 与 b_2，其中 b_1 与源代码 s 相似，而 b_2 与源代码 s 不相似。基于这些信息，可以构建一个三元组 $< s, b_1, b_2 >$，并将相似度检测任务定义为：

$$\ell(\theta) = \sum_{<s,b_1,b_2>\in D} max\left(0, \beta - sim\left(\psi_s, \Upsilon_{b_1}\right) + sim\left(\psi_s, \Upsilon_{b_2}\right)\right) \#$$

（17）

其中，$sim(x,y)$ 为模型训练中使用的相似度度量，它通过采用余弦相似度 Cosine 来评估特征向量 x 与 y 之间的相似性，这些特征向量由代码表示学习模型生成；θ 表示模型训练过程中的待优化参数；D 是构建为源 - 二进制代码对的三元组形式的训练数据集；y 和 g 分别是为源代码和二进制代码各模态特征设计的表示学习模型的统称；b 是一个用以区分相似对与不相似对的非常小的常数，论文中设置为 0.06。

6 结语

论文采用 Joern 和 IDA Pro 工具从源代码和二进制代码中提取控制流图，将逻辑调用信息加入特征集，提高相似性检测准确率，克服传统方法的局限。通过语义编码网络，论文从代码的序列 token、字符字面量和控制流图中提取特征，生成融合的代表向量，用余弦相似度评估代码对的相似性。针对编译条件变化导致的二进制代码差异，论文提出基于对比学习的二进制代码预训练模型，与 CodeBERT 结合应用于跨形态代码相似性检测，效果良好。

论文在跨形态代码相似性研究中仍有不足，未来可继续深耕以下方面：

①引入注意力机制，明确各模态的贡献比重；

②探索将整数信息作为新模态加入相似性检测；

③开发跨语言的源代码语义编码网络，专注代码语义，实现更有效的相似性检测。

参考文献

[1] Andrew G, Arora R, Bilmes J, et al. Deep canonical correlation analysis[C]. International conference on machine learning. PMLR,2013(6):1247-1255.

[2] Afham M, Dissanayake I, Dissanayake D, et al. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,2022(6):9902-9912.

[3] Zhou J, Liu L, Wei W, et al. Network representation learning: from preprocessing, feature extraction to node embedding[J]. ACM Computing Surveys (CSUR),2022,55(2):1-35.