计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(12):253−260 [doi: 10.15888/j.cnki.csa.009303]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于词序嵌入的二进制基本块相似性检测①

李　涛, 王金双, 周振吉

(陆军工程大学 指挥控制工程学院, 南京 210007)
通信作者: 王金双, E-mail: submitarticle23@sohu.com

摘　要: 神经机器翻译技术能够自动翻译多种语言的语义信息, 已被应用于跨指令集架构的二进制代码相似性检测, 并取得了较好的效果. 将汇编指令序列当作文本序列处理时, 指令顺序关系很重要. 进行二进制基本块级别相似性检测时, 神经网络使用位置嵌入来对指令位置进行建模. 然而, 这种位置嵌入未能捕获指令位置之间的邻接、优先等关系. 针对该问题, 本文使用指令位置的连续函数来建模汇编指令的全局绝对位置和顺序关系, 实现对词序嵌入的泛化. 首先使用 Transformer 训练源指令集架构编码器; 然后使用三元组损失训练目标指令集架构编码器, 并微调源指令集架构编码器; 最后使用嵌入向量之间欧氏距离的映射表示基本块之间的相似程度. 在公开数据集 MISA 上的实验表明, *P*@1 评价指标达到 69.5%, 比对比方法 MIRROR 提升了 4.6%.

关键词: 二进制基本块; 相似性检测; 跨指令集架构; 神经机器翻译; 词序嵌入

引用格式: 李涛,王金双,周振吉.基于词序嵌入的二进制基本块相似性检测.计算机系统应用,2023,32(12):253–260. http://www.c-s-a.org.cn/1003-3254/9303.html

Binary Basic Block Similarity Detection Based on Word Order Embeddings

LI Tao, WANG Jin-Shuang, ZHOU Zhen-Ji

(Command and Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China)

Abstract: Neural machine translation technology can translate the semantic information of multiple languages automatically. Therefore, it has been applied to binary code similarity detection of cross-instruction set architecture successfully. When the sequences of assembly instructions are treated as sequences of textual tokens, the order of instructions is important. When binary basic block-level similarity detection is performed, the neural networks model instruction positions with position embeddings, but it failed to reflect the ordered relationships (e.g., adjacency or precedence) between instructions. To address this problem, this study uses a continuous function of instruction positions to model the global absolute positions and ordered relationships of assembly instructions, achieving the generalization of word order embeddings. Firstly, the source instruction set architecture (ISA) encoder is constructed by Transformer. Secondly, the target ISA encoder is trained by triplet loss, and the source ISA encoder is fine-tuned. Finally, the Euclidean distances between embedding vectors are mapped to [0,1], which are used as the similarity metrics between basic blocks. The experimental results on the public dataset MISA show that the evaluation metric *P*@1 of this study is 69.5%, which is 4.6% higher than the baseline method MIRROR.

Key words: binary basic block; similarity detection; cross-instruction set architecture; neural machine translation; word order embedding (WOE)

二进制代码相似性检测技术被广泛应用于漏洞搜索[1–4]、代码复用检测[5]、恶意软件检测[6]等领域. 随着物联网技术的发展, 越来越多的软件被部署到不同指令集架构 (instruction set architecture, ISA)(如 ARM、

① 收稿时间: 2023-05-05; 修改时间: 2023-06-06; 采用时间: 2023-06-26; csa 在线出版时间: 2023-09-21
CNKI 网络首发时间: 2023-09-22

MIPS、RISC-V 等) 的平台上运行. 由于函数调用约定、CPU 寄存器和内存访问策略在不同 ISA 中具有较大差异, 使得相同的源代码经过编译后得到的二进制代码也非常不同, 给跨 ISA 的相似性检测带来了较大的挑战[7].

神经机器翻译 (neural machine translation, NMT) 的思想已被用于跨 ISA 的相似性检测任务, 文献 [8,9] 使用 LSTM[10]、Transformer[11]等神经网络训练跨 ISA 的指令序列编码器, 较传统方法展现出了良好的准确性和扩展性. 但是检测方法[9]使用的神经网络在特征级别添加位置嵌入, 且假设单个指令位置是独立的, 并未考虑相邻指令之间关系. 然而, 汇编基本块中指令的全局绝对位置及其内部顺序和相邻关系在汇编代码中具有重要作用, 如: PUSH 和 POP 指令通常固定搭配使用并且通常分布于汇编基本块序列的两端; 跳转指令如 JMP 通常在序列中处于 MOV 和 ADD 等指令助记符之后.

针对上述问题, 将指令嵌入拓展为以位置为变量的连续函数的词序嵌入 (word order embedding, WOE). 词序嵌入能够根据同一指令的向量推导出位置的相对距离, 也能推导出不同指令间的位置关系. 基于 NMT 的思想, 将词序嵌入后的指令序列输入 Transformer 神经网络中, 源 ISA 汇编基本块序列在训练过程中被转换为目标 ISA 汇编基本块序列, 得到能够提取源 ISA 汇编基本块语义特征的编码器; 然后通过三元组损失训练目标 ISA 编码器和源 ISA 编码器; 两阶段训练得到的跨 ISA 编码器用于相似性检测任务.

本文实现了跨 ISA 二进制基本块粒度的相似性检测方法 WOE-MIRROR. 实验结果表明, 在相似基本块搜索任务中, *P*@1 精度 (目标在前 1 个结果中的准确率) 达到 69.5%, 比对比方法 MIRROR 提升了 4.6%, 证明了本文检测方法的有效性.

1 相关研究

1.1 二进制代码相似性检测

基于静态分析的检测方法主要通过人工选取二进制的统计特征、结构特征等. 统计特征主要包括按类别划分的指令统计数目; 结构特征主要包括函数的控制流图 (control flow graph, CFG)[12,13]、数据流图 (data flow graph, DFG)[2]、属性控制流图 (attributed control flow graph, ACFG)[13] 等. 基于统计特征的方法依赖于测试者的领域知识, 难以精确捕获二进制代码语义特征, 因此这些方法往往具有较低的精度.

自然语言处理技术已应用于二进制分析领域, 优势在于神经网络能够自动学习代码序列的语义特征, 并且避免了复杂的图匹配算法或者动态执行开销.

基本块粒度的检测任务可以作为函数级别相似性检测的子任务, 也可以直接用于二进制分析任务. 文献 [14]使用图神经网络学习二进制函数 ACFG 特征; jTrans[15]将跳转指令预测任务融合 BERT[16]的预训练方法, 训练函数级的汇编代码的程序语言模型. 但是这些模型是为单一 ISA 的函数粒度检测而设计, 难以应用于跨 ISA 的二进制基本块相似性检测.

1.2 神经机器翻译

NMT 技术能够以端到端的方式学习输入文本到输出文本的映射, 常用于不同词表的文本转化任务. 使用的神经网络结构为 Seq2Seq 结构, 主要包含两个部分: 编码器和解码器, 一个用于输入源文本序列, 另一个用于生成翻译后的输出文本. 编码器-解码器架构 (以 Transformer 为例) 如图 1 所示.

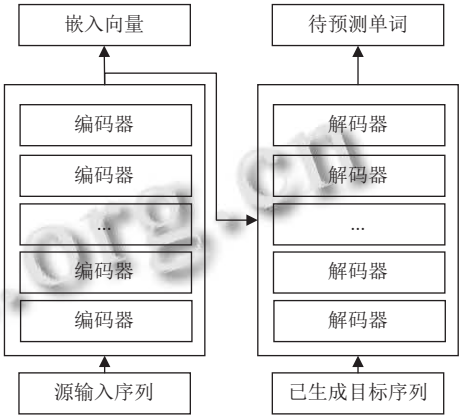

图 1 编码器-解码器框架

编码器能够将文本序列嵌入到向量空间, 由解码器对输出序列进行逐词预测, 实现跨语言翻译, 所以 NMT 技术可用于解决跨 ISA 的二进制代码相似性检测. 并且使用 Seq2Seq 神经网络 (如 LSTM[10]、Transformer[11]等) 的跨指令集的检测方法已取得了比传统的静态相似性检测方法更高的准确度[8,9]. LSTM 模型在长序列中表现不佳, 而基于注意力机制的 Transformer 虽然在长文本序列建模中效果较好, 但是绝对位置编码难以区分指令助记符和操作数对模型的贡献度, 使得检测模型在学习指令序列的位置信息上存在欠缺.

1.3 位置编码

自 Transformer 出现以来, 一些自然语言处理领域的工作研究位置编码[17−20]. 相对位置编码和绝对位置编码以不同的方式向网络中加入位置信息, 以提升网络模型对于文本位置信息的学习能力. 针对不同的任务, 不同位置编码的效果具有差异性.

Vaswani[11]采用编码器-解码器结构, 将基于变频正弦波的位置编码添加到第 1 层之前的编码器和解码器的输入中, 正弦位置编码可以让模型学习相对位置来帮助模型推广到训练过程中神经网络无法看到的序列长度. BERT[16]、GPT[20]等模型采用训练式的相对位置编码, 通过初始化一个固定维度的矩阵作为位置向量, 并且矩阵可以随着训练过程进行更新.

Show[18]提出了一种相对位置编码, 通过扩展自注意力机制, 将注意力矩阵中的位置向量转化为二维位置信息, 因为对序列进行了截断, 所以这种方式能够使用有限个位置编码表示任意长度的相对位置; XLNET[19]、T5[21]等则将相对位置加入注意力矩阵中.

2 模型的框架和训练流程

本文的二进制基本块相似性检测模型共包括 3 个模块: 数据预处理模块、语义嵌入模块和相似性计算模块, 如图 2 所示. ① 数据预处理模块主要是对二进制文件进行反汇编和数据规范化处理; ② 语义嵌入模块主要对汇编指令序列进行指令和位置信息编码, 然后经过词序嵌入模块后, 输入编码器进行语义嵌入, 得到汇编基本块的语义向量; ③ 相似性计算模块通过计算语义向量之间的欧氏距离后用于衡量二进制基本块的语义相似程度.

为了得到汇编基本块嵌入向量, 需要训练一个源 ISA 编码器和一个目标 ISA 编码器, 选用代表性的 x86 指令集和 ARM 指令集作为检测对象. 通过训练两个不同 ISA 的语义编码器, 可以将源指令序列和目标指令序列映射到同一个向量空间.

模型训练流程包含 3 个阶段: 数据预处理阶段, 源编码器训练阶段, 目标编码器训练阶段, 如图 3 所示. Seq2Seq 的神经网络能够将源 ISA 的汇编语言翻译成目标 ISA 的汇编语言, 经由编码器得到的多维向量包含源 ISA 的语义特征; 为了得到目标 ISA 的编码器, 通过三元组损失训练目标 ISA 编码器和源 ISA 编码器. 三元组损失中负样本的加入可以推动样本的语义向量在空间中的分布更加均匀.

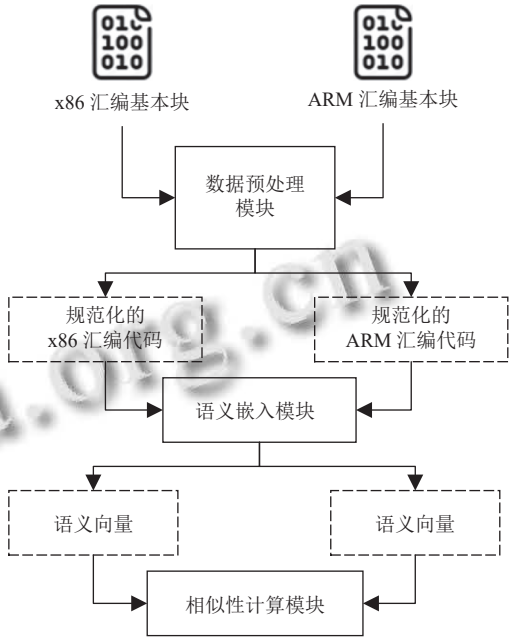

图 2 WOE-MIRROR 相似性检测框架

2.1 数据预处理阶段

模型训练阶段需要大量的相似基本块对, 本文使用开源数据集 MISA[9]中的 x86-ARM 汇编基本块相似对. 该数据集的生成方法为: 修改 LLVM, 为 LLVM IR 中的每个基本块分配唯一的 ID, 面向不同指令集架构编译之后 (如 x86 和 ARM 等), 所生成的汇编基本块只要源自同一 ID, 相互之间就构成相似对.

由于汇编代码中包含大量的指针、地址、函数名等信息, 这些信息将产生不可预测的词表, 直接将原始的指令序列输入机器学习模型进行训练极易引发词汇表溢出问题, 因此需要指令规范化处理, 如图 4 所示. 在指令规范化步骤中将汇编代码中的常量分为 5 类: 立即数、地址、变量名、函数名和基本块标签; 将 x86 指令集的寄存器分为 14 类: 指针寄存器、浮点寄存器、4 类通用寄存器、4 类数据寄存器和 4 类地址寄存器; 将 ARM 指令集寄存器分为通用寄存器和指针寄存器 2 类.

三元组中的负样本采用随机采样和困难负样本采样结合的方式生成[9,22,23]. 困难负样本通过正样本与随机筛选的相同 ISA 基本块代码计算相似性得分, 相似性排序高的作为困难负样本, 向量距离如式 (1) 所示:

$$D(E_1,E_2)=\sqrt{\sum_{i=1}^{d}(e_{1i}-e_{2i})^2} \tag{1}$$

其中, E_1, E_2为指令序列嵌入, $e_{1i}\in E_1$, $e_{2i}\in E_2$, d 为嵌入维度. $D(E_1,E_2)$的值越小, 表明两个基本的相似性越高. 为了量化相似性得分, 通过将欧氏距离映射到一个固定的区间 [0, 1]上, 基本块$Block_1$和$Block_2$的相似性得分定义为:

$$Sim(Block_1,Block_2)=\exp\left(-\frac{D(E_1,E_2)}{d}\right) \tag{2}$$

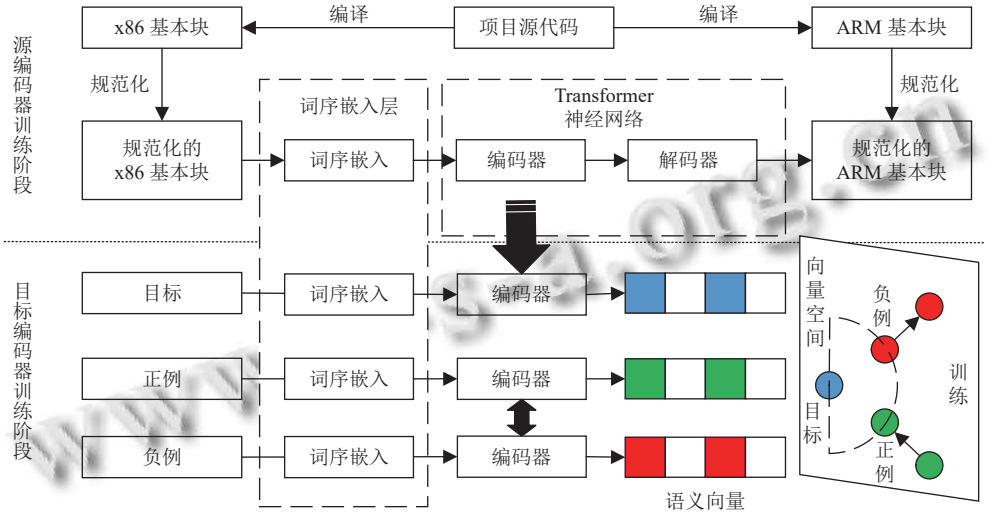

图 3 WOE-MIRROR 训练流程

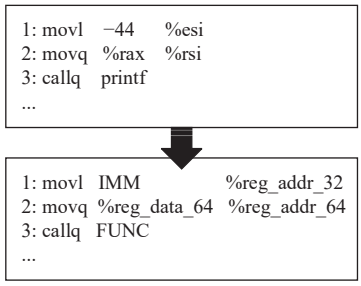

图 4 指令规范化

2.2 词序嵌入

为了得到源 ISA 编码器, 需要对数据进行词表映射生成指令编码和位置编码, 然后输入神经网络. 文献 [9]使用原始 Transformer 网络, 将指令序列映射到词表进行编码后, 以指令嵌入和词序嵌入以求和的方式输入到神经网络, 形式如下:

$$f(l,p)=f_{\text{ins}}(l)+f_p(p) \tag{3}$$

其中, l为指令在词汇表中的索引, p为指令在序列中的位置信息. 指令嵌入$f_{\text{ins}}:\mathbb{N}\to\mathbb{R}^D$表示将指令嵌入到一个 D 维的向量空间; 位置嵌入$f_p:\mathbb{N}\to\mathbb{R}^D$表示将离散的位置索引嵌入到向量空间. 指令嵌入映射f_{ins}和位置嵌入映射f_p都只取自然数值作为指令索引或位置索引, 因此对单个指令位置信息的嵌入向量进行独立训练, 位置嵌入f_p的独立训练使得指令之间的有序关系无法被模型捕获.

受 Wang 等人[24]的启发, 为了建模不同位置指令的相对关系, 将位置嵌入拓展为一个以位置为变量的连续函数, 不同位置的指令嵌入可以在连续函数中相互关联. 指令嵌入函数的形式如下:

$$f(l,p)=h_l(p)\in\mathbb{R}^D \tag{4}$$

其中, h_l是$h_{\text{ins}}(l)\in(F)^D$的简称, $h_{\text{ins}}(\cdot):\mathbb{N}\to(F)^D$表示指令索引到 D 维函数的映射, $h_l(p)$表示位置索引p的函数. 此时位置为p的指令向量i_l可以被拓展成一个 D 维的向量.

$$[h_{l,1}(p),h_{l,2}(p),\cdots,h_{l,D}(p)]\in\mathbb{R}^D \tag{5}$$

其中, $h_{l,d}(\cdot)\in F:\mathbb{N}\to\mathbb{R},d\in\{1,2,\cdots,D\}$是关于位置索引$p$的函数, 指令嵌入每个维度的向量均与位置索引$p$相关. 将词嵌入$h_{l,D}(p)$目标域拓展到复数域[24], 此时指令嵌入被定义为一个将指令索引l和位置索引p到复数域的映射, 公式为$f(l,p)=h_l(p)=r_l e^{i(\omega_l p+\theta_l)}$; 通过限制$r_l=1$, 可以避免值域为实数域时指令嵌入函数收敛到 0 或者与位置信息无关两种情况, 此时实数域可以

看作复数域的子集. 指令嵌入函数$f(l,p)$表示为:

$$\left[r_{l,1}\mathrm{e}^{i(\omega_{l,1}p+\theta_{l,1})},\cdots,r_{l,2}\mathrm{e}^{i(\omega_{l,2}p+\theta_{l,2})},\cdots,r_{l,D}\mathrm{e}^{i(\omega_{l,D}p+\theta_{l,D})}\right] \tag{6}$$

其中, 每个坐标d具有独立且可学习的振幅$r_{l,d}$、周期$p_{l,d}=\dfrac{2\pi}{\omega_{l,d}}$和初始相位矢量$\theta_{l,d}$, 指令嵌入的振幅$r_{l,d}$取决于指令向量$i_l$和坐标$d$, 而不取决于指令的位置. 对基本块序列进行词序嵌入后, 输入 Transformer 用于后续编码器训练.

2.3 源编码器训练阶段

本文把独立指令当作单词, 按照地址序列化的汇编基本块当作句子. 编码得到 x86 指令序列$S=(s_1,s_2,\cdots,s_n)$和 ARM 指令序列$T=(t_1,t_2,\cdots,t_n)$后, 输入第 2.2 节所述的词序嵌入模块. 将 x86 指令序列输入 Transformer 神经网络的编码器进行正向传播, 可以从源编码器的输出中得到具有 x86 基本块语义信息的上下文矩阵$C=\{c_i\}\in\mathbb{R}^{d\times n}$, 其中 n 表示 x86 基本块序列长度, d 表示向量维度. 上下文矩阵C为固定维度的向量, 作为汇编指令序列的语义嵌入向量.

编码器的输出作为解码器的输入, 在解码器中正向传播, 以预测生成目标指令. 在解码器部分, 将初始位置 1 到预测位置$l-1$的目标指令序列$T=(t_1,t_2,\cdots,t_{l-1})$输入解码器, 用于输出下一个指令$y_l$的概率分布$y_l\in\mathbb{R}^{|V_{\text{target}}|}$, 所用损失函数如下:

$$Loss=-\sum_{l=1}^{m}\sum_{j=1}^{|V_{\text{target}}|}\hat{y}_{lj}\log\left(y_{lj}\right) \tag{7}$$

其中, $\hat{y}_l\in\mathbb{R}^{|V_{\text{target}}|}$是目标指令$t_l$的独热编码. 本文使用 Adam[25]梯度下降算法计算每个参数的学习率使训练损失最小化. 训练得到的 x86 编码器用于 ARM 编码器训练阶段.

2.4 目标编码器训练阶段

目标编码器训练阶段是为了得到两种架构的编码器进行后续相似性分析任务. 得到 x86 源编码器后, 本文通过构建<目标、正例、负例>的三元组训练 ARM 语义编码器和 x86 语义编码器, 训练的目标是目标样本和正例样本在向量空间的距离比目标样本和负例样本间的距离更小.

三元组中正例和负例来源于同一指令集架构, 目标样本来自于另一个指令集架构, 通过第 2.1 节所述方法进行采样, 三元组损失训练过程如图 5 所示.

三元组损失函数能够优化目标和正例与目标和负例之间的距离差范围, 所用损失函数如下:

$$Loss=\max\left\{0,\alpha+D\left(E_a,E_p\right)-D(E_a,E_n)\right\} \tag{8}$$

其中, α表示三元组样本间的距离, 表示学习目标为正例样本到目标之间的距离和负例样本到目标之间距离差至少为α. 目标编码器训练结束后, 可以使用编码器对两种 ISA 汇编基本块语义嵌入后进行相似度计算, 如图 6 所示.

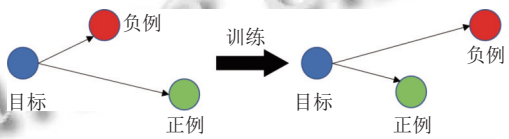

图 5 三元组损失

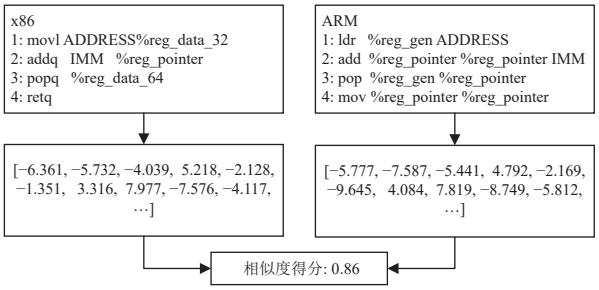

图 6 基本块相似度计算

3 实验验证

3.1 实验环境

将 WOE-MIRROR 和 MIRROR 模型均部署于如下环境中进行训练评估: 一台搭载 Ubuntu Server 18.04 LTS 64 位操作系统的服务器, 硬件配置包括: CPU 为 1 个 Intel(R) Xeon(R) Gold 6133 CPU @ 2.50 GHz, 显卡为 1 块 Tesla V100 GPU, 内存大小为 40 GB.

3.2 数据集和训练参数设置

用于训练和评估的数据集为开源数据集 MISA, 共包含 1 122 171 个语义相等的 x86 架构和 ARM 架构二进制基本块对. 该数据集由 5 个不同领域的 C/C++开源项目使用 4 种不同的优化选项 (O0–O3) 编译得到, 具体如表 1 所示. 本文从数据集中随机选取 240 000 个相似基本块对用于源编码器训练, 随机选取 60 000 个相似对作为测试集; 通过第 2.1 节所述负样本生成方法生成三元组用于目标编码器训练, 随机样本和困难负

样本比例为 2:1. 在训练过程中使用和 MIRROR 相同的训练数据集和超参数, α 设置为 140.

表 1 MISA 数据集

项目名称	版本号
Binutils	2.30
Coreutils	8.29
FFmpeg	n3.2.13
OpenSSL	1.1.1b
Redis	5.0.5

3.3 训练过程比较

源编码器训练阶段使用交叉熵损失, 交叉熵损失能够度量模型预测结果和实际情况之间的差异; 训练阶段的三元组损失则可以计算目标和正负样本之间的距离损失, 损失越低表明模型对于负样本的区分度越好, 源编码器训练阶段和目标编码器训练阶段的轮数均为 20, 为图 7–图 9 的横坐标.

图 7、图 8 所示曲线表明, 在源编码器训练阶段中 WOE-MIRROR 的损失曲线较低, 精度曲线较高, 即在损失和精度表现上均优于 MIRROR.

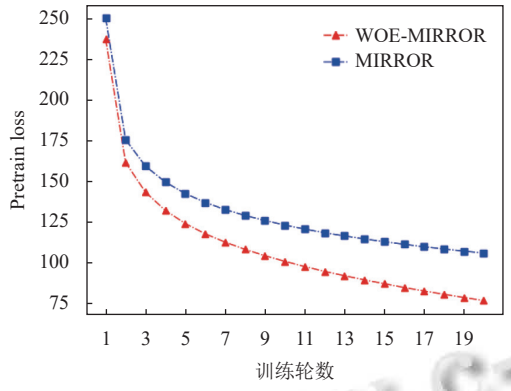

图 7 预训练过程损失值对比

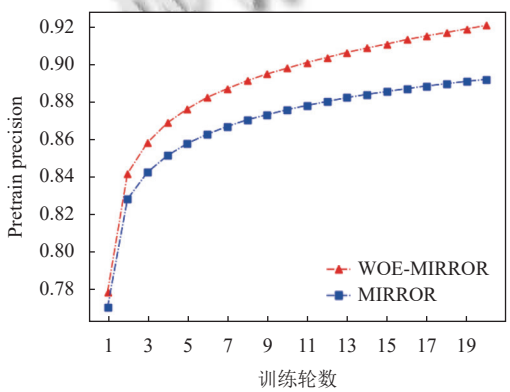

图 8 预训练过程准确率对比

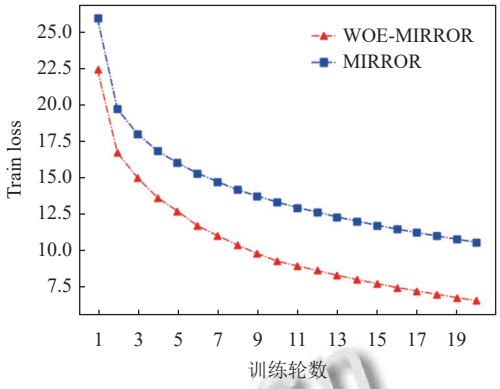

图 9 训练过程损失值对比

图 9 所示曲线表明, 在目标编码器训练阶段, WOE-MIRROR 的三元组损失曲线比 MIRROR 更低, 即加入词序嵌入模块后对模型训练具有较强的促进作用, 表明本方法对于指令语义特征的差异性具有更好的区分能力.

3.4 相似代码检索

为了衡量模型的相似性搜索效果, 通过在离线状态下评估模型在测试集搜索目标基本块的能力. 相似基本块搜索任务为给定一个相似基本块对 Q 和 f_q, 将从样本池中随机筛选的基本块和查询基本块 Q 计算相似性评分并进行排序 $\{f_1, f_2, \cdots, f_k | f_i \in p\}$, 真值样本 f_q 在排名前 n 的概率为 $P@N$, 得分越高说明模型的检测能力越强.

由于现有跨 ISA 二进制基本块相似性检测研究中源码不开放和数据集不统一等情况, 同时文献 [9]已经证明使用 Transformer 的模型检测效果优于 LSTM[8], 所以本文选择文献 [12,13]中基本块特征提取方法作为对比方法进行比较. 选择的基本块属性表 2 所示.

表 2 Genius 基本块特征

类型	特征名称
基本块级属性	字符串常量数目
	数值常量数目
	转移指令数目
	调用指令数目
	指令数目
	算数指令数目

$P@N$ 任务共包含两部分: x86-ARM 任务和 ARM-x86 任务. x86-ARM 任务即给定 x86 基本块, 在目标中寻找目标 ARM 基本块; ARM-x86 任务即给定 ARM 基本块寻找目标 x86 基本块. 将搜索池设置为 100, 即在测试集中随机选择 100 个样本用于与查询样本计算相似性并排序.

P@N 任务实验结果如表 3、表 4 所示. 其中, Genius-b 表示文献 [12]中手工提取基本块特征的方法. 结果显示, 本模型在 *P@N* 指标均优于对比模型.

表 3 x86-ARM *P@N* 结果比较 (%)

模型	x86-ARM任务		
	P@1	*P*@3	*P*@10
Genius-b	21.6	32.5	49.6
MIRROR	71.3	82.2	89.5
WOE-MIRROR	**75.6**	**85.4**	**92.3**

表 4 ARM-x86 *P@N* 结果比较 (%)

模型	ARM-x86任务		
	P@1	*P*@3	*P*@10
Genius-b	19.5	31.6	46.2
MIRROR	64.9	78.2	86.4
WOE-MIRROR	**69.5**	**82.1**	**89.8**

3.5 相似性得分可视化

从测试数据集中随机选取 10 个语义相似的跨 ISA 基本块对, 输入模型后得到语义向量, 计算相似度得分后进行可视化展示, 如图 10 所示. 结果显示, 对角线的向量相似性评分高于其他行列的得分, 表明模型具备对二进制基本块进行语义特征提取能力并且产生了较好的相似性区分度.

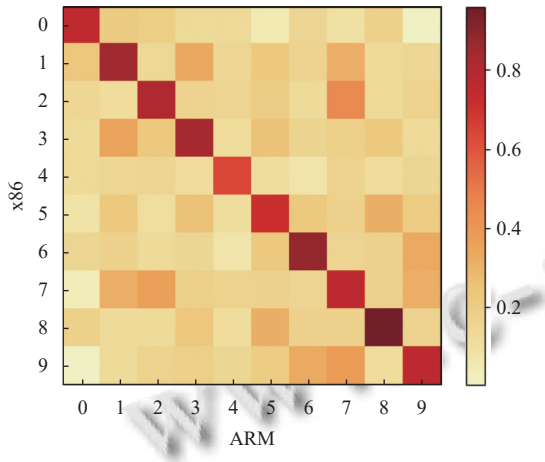

图 10 向量相似度可视化

4 结论与展望

本文以位置为变量的函数编码二进制汇编指令序列, 通过 Transformer 神经网络和三元组损失网络训练跨 ISA 的语义编码器. 实验表明, 以位置为变量的词序嵌入能够提升模型对二进制汇编基本块的语义嵌入能力. 未来的工作将在提高模型的拓展性等方面进行研究.

参考文献

1 David Y, Partush N, Yahav E. FirmUp: Precise static detection of common vulnerabilities in firmware. ACM SIGPLAN Notices, 2018, 53(2): 392–404. [doi: 10.1145/3296957.3177157]

2 Gao J, Yang X, Fu Y, *et al*. VulSeeker: A semantic learning based vulnerability seeker for cross-platform binary. Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering. Montpellier: IEEE, 2018. 896–899.

3 Gao J, Yang X, Fu Y, *et al*. Vulseeker-pro: Enhanced semantic learning based binary vulnerability seeker with emulation. Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Lake Buena Vista: ACM, 2018. 803–808.

4 陈斌, 刘胜利, 胡安祥, 等. 基于神经机器翻译的二进制函数相似性检测方法. 信息工程大学学报, 2021, 22(6): 675–682. [doi: 10.3969/j.issn.1671-0673.2021.06.007]

5 Luo LN, Ming J, Wu DH, *et al*. Semantics-based obfuscation-resilient binary code similarity comparison with applications to software and algorithm plagiarism detection. IEEE Transactions on Software Engineering, 2017, 43(12): 1157–1177. [doi: 10.1109/TSE.2017.2655046]

6 Alrabaee S, Shirani P, Wang LY, *et al*. *FOSSIL*: A resilient and efficient system for identifying FOSS functions in malware binaries. ACM Transactions on Privacy and Security, 2018, 21(2): 8.

7 于颖超, 甘水滔, 邱俊洋, 等. 二进制代码相似度分析及在嵌入式设备固件漏洞搜索中的应用. 软件学报, 2022, 33(11): 4137–4172. [doi: 10.13328/j.cnki.jos.006540]

8 Zuo F, Li XP, Young P, *et al*. Neural machine translation inspired binary code similarity comparison beyond function pairs. Proceedings of the 26th Annual Network and Distributed System Security Symposium. San Diego: The Internet Society, 2019.

9 Zhang XC, Sun WJ, Pang JM, *et al*. Similarity metric method for binary basic blocks of cross-instruction set architecture. Proceedings of the 2020 Workshop on Binary Analysis Research. San Diego, 2020. 23–26.

10 Hochreiter S, Schmidhuber J. Long short-term memory. Proceedings of the 2015 International Conference on Machine Learning. PMLR, 2015. 1604–1612.

11 Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach: Curran

Associates Inc., 2017. 6000–6010.

12 Feng Q, Zhou RD, Xu CC, *et al*. Scalable graph-based bug search for firmware images. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016. 480–491.

13 Xu XJ, Liu C, Feng Q, *et al*. Neural network-based graph embedding for cross-platform binary code similarity detection. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas: ACM, 2017. 363–376.

14 Massarelli L, Di Luna GA, Petroni F, *et al*. Investigating graph embedding neural networks with unsupervised features extraction for binary analysis. Proceedings of the 2nd Workshop on Binary Analysis Research (BAR). 2019. 1–11.

15 Wang H, Qu WJ, Katz G, *et al*. jTrans: Jump-aware transformer for binary code similarity detection. Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM, 2022. 1–13.

16 Devlin J, Chang MW, Lee K, *et al*. BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis: ACL, 2018. 4171–4186.

17 Huang ZH, Liang D, Xu P, *et al*. Improve transformer models with better relative position embeddings. Proceedings of the 2020 Findings of the Association for Computational Linguistics: EMNLP 2020. ACL, 2020. 3327–3335.

18 Shaw P, Uszkoreit J, Vaswani A. Self-attention with relative position representations. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. New Orleans: ACL, 2018. 464–468.

19 Yang ZL, Dai ZH, Yang YM, *et al*. XLNet: Generalized autoregressive pretraining for language understanding. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Vancouver, 2019. 517.

20 Radford A, Wu J, Child R, *et al*. Language models are unsupervised multitask learners. OpenAI Blog, 2019, 1(8): 9.

21 Raffel C, Shazeer N, Roberts A, *et al*. Exploring the limits of transfer learning with a unified text-to-text Transformer. The Journal of Machine Learning Research, 2020, 21(1): 140.

22 Chen WH, Chen XT, Zhang JG, *et al*. Beyond triplet loss: A deep quadruplet network for person re-identification. Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition. Honolulu: IEEE, 2017. 1320–1329.

23 Xiao QQ, Luo H, Zhang C. Margin sample mining loss: A deep learning based method for person re-identification. arXiv:1710.00478, 2017.

24 Wang B, Donghao Z, Christina L, *et al*. Encoding word order in complex embeddings. Proceedings of the International Conference on Learning Representations (ICLR). 2020.

25 Kinga D, Adam JB. A method for stochastic optimization. Proceedings of the International Conference on Learning Representations (ICLR). 2015.

(校对责编: 牛欣悦)