计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.20220014
59(9):1914-1928,2022

基于深度学习的数据竞争检测方法

张 杨 乔 柳 东春浩 高鸿斌
(河北科技大学信息科学与工程学院 石家庄 050018)
(zhangyang@hebust.edu.cn)

Deep Learning Based Data Race Detection Approach

Zhang Yang, Qiao Liu, Dong Chunhao, and Gao Hongbin
(*College of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang* 050018)

Abstract Existing approaches for deep-learning-based data race detection are suffering from the issues of single feature extraction and low accuracy. To improve the state-of-the-art, a novel approach called DeleRace is proposed to detect data race based on deep learning model. Firstly, DeleRace extracts instruction-level, method-level, and file-level features from a variety of real-world applications based on static analysis tool WALA. All these features are transformed by word vectorization to build the training dataset. Secondly, ConRacer, as an existing data race tool, is employed to identify the real race. Based on this tool, those positive samples in the training dataset is labelled. To further optimize the dataset, DeleRace leverages SMOTE algorithm to distribute both positive samples and negative ones in balance. Finally, CNN-LSTM model is constructed and a classifier is trained to detect data race. In the experimentation, a total of 26 real-world applications is selected from different fields in DaCapo, JGF, IBM Contest and PJBench benchmark suites. The experimental results show that the accuracy of DeleRace is 96.79% which is 4.65% higher than existing deep-learning-based approaches. Furthermore, the performance of DeleRace is compared with that of both dynamic tools (such as Said and RVPredict) and static tools (such as SRD and ConRacer), which demonstrates the effectiveness of DeleRace.

Key words data race; concurrent program; deep learning; feature extraction; CNN-LSTM model

摘 要 针对目前已有的基于深度学习的数据竞争检测方法提取特征单一和准确率低的问题,提出一种基于深度学习的数据竞争检测方法 DeleRace,该方法首先利用程序静态分析工具 WALA 从多个实际应用程序中提取指令、方法和文件等多个级别的特征,对其向量化并构造训练样本数据;然后通过 ConRacer 工具对真实数据竞争进行判定进而标记样本数据,采用 SMOTE 增强算法使正负数据样本分布均衡化;最后构建并训练 CNN-LSTM 深度神经网络进行数据竞争检测.从 DaCapo,JGF,IBM Contest,PJBench 基准测试程序套件中分别选取 26 个不同应用领域的基准测试程序进行训练数据样本抽取和数据竞争检测,结果表明 DeleRace 的数据竞争检测准确率为 96.79%,与目前已有的基于深度学习的检测方法 DeepRace 相比提升了 4.65%.此外还将 DeleRace 与已有的动态数据竞争检测工具

收稿日期:2022-01-04;**修回日期**:2022-04-24
基金项目:国家自然科学基金项目(61440012);河北省高等学校科学研究计划重点项目(ZD2019093);河北省科技支撑计划项目(16210312D);河北省研究生创新能力培养资助项目(CXZZSS2022081)
This work was supported by the National Natural Science Foundation of China (61440012), the Key Scientific Research Project of Hebei Education Department (ZD2019093), the Scientific Support Project of Hebei Province (16210312D), and the Innovative Ability Foundation for Graduates of Hebei Province (CXZZSS2022081).

(Said 和 RVPredict)和静态数据竞争检测工具(SRD 和 ConRacer)进行比较,验证了 DeleRace 的有效性.

关键词　数据竞争;并发程序;深度学习;特征抽取;CNN-LSTM 模型

中图法分类号　TP311

数据竞争[1]是指 2 个或多个线程同时访问 1 个内存位置并且至少有 1 个线程执行写操作.数据竞争是目前最常见的并发缺陷之一,它是一种典型的运行时故障,通常在特定的并发执行环境中发生,难以被检测,它的存在会给程序运行带来潜在的风险,严重时会导致程序无法正常运行甚至崩溃,造成无法估量的损失,因此迫切需要对数据竞争检测问题进行研究.

数据竞争检测一直是国内外并发缺陷研究领域的热点问题之一,很多学者对数据竞争检测问题进行了研究,所采用的方法包括基于动态程序分析的检测方法、基于静态程序分析的检测方法、动静结合的检测方法[2-4].基于动态程序分析的检测方法在程序运行过程中,通过监控程序执行路径和内存读写访问等方式检测数据竞争的发生,这种检测方式的优点在于误报率较低,缺点是数据竞争的漏报率较高,而且检测过程开销较大.已有的动态数据竞争检测工具有 Said[5],RVPredict[6],SlimFast[7] 等.与动态数据竞争检测方法不同,静态数据竞争检测在源代码或中间代码层次展开,通过分析程序中变量的读写访问,辅助以各种静态程序分析技术(如发生序分析、别名分析、逃逸分析等)进行数据竞争检测.这种检测方式的优点在于可以在程序运行之前排除相关问题,不仅开销较小,而且检测较为全面,漏报率较低;缺点在于仅在代码层面进行分析而没有真正运行程序,可能会导致很多误报,已有的静态检测工具包括 RELAY[8],Elmas[9],SRD[10] 等.此外,为了弥补以上 2 种方式各自的不足,有些研究人员也尝试将动态分析和静态分析 2 种检测方法结合起来,以此提高检测的整体效率,常用的动静结合的检测工具有 RaceTracker[11] 和 AsampleLock[12] 等.

近年来,随着机器学习和深度学习技术的发展和广泛应用,一些研究人员开始将相关技术应用于数据竞争检测.在国内,孙家泽等人[13-14]提出一种基于机器学习的数据竞争检测方法,该方法使用随机森林模型,收集指令级别数据,进行数据竞争检测,而且他们还基于 Adaboost 模型进行语句级并发程序数据竞争检测,该方法的准确率可达 92%.在国外,Tehrani 等人[15]通过提取文件级别的特征来构建数据竞争训练数据集,他们提出一种基于卷积神经网络(convolutional neural network, CNN)的数据竞争检测方法,实验表明该方法检测的准确率在 83%～86%.

从目前的研究现状来看,一些研究人员从程序分析的角度开展数据竞争检测研究,另一些研究人员将程序作为语料库,使用机器学习和深度学习方法开展研究,虽然已有的检测方法取得了一定的成效,但仍存在 3 个方面的问题亟需进一步研究完善:

1) 目前已有工作所使用的学习模型主要依赖于深度学习中 CNN 模型和机器学习中随机森林模型,模型还有待于优化,准确率还有提升的空间.

2) 在构建数据集时,现有的基于深度学习的数据竞争检测工具[15]仅应用了 3 个不同的基准测试程序,所提取的数据集样本个数较少,在输入到深度学习模型时会导致检测精度下降.

3) 在特征抽取时,仅提取指令或文件等级别的某一方面的特征,无法充分反映数据竞争的真实情况.

针对目前研究存在的问题,本文提出一种基于深度学习的数据竞争检测方法 DeleRace(deep-learning-based data race detection).该方法首先使用程序静态分析工具 WALA[16] 从多个实际应用程序中提取指令、方法和文件级别中多个代码特征,对其向量化并构造训练样本数据;然后通过 ConRacer[17] 工具对真实数据竞争进行判定进而标记样本数据,并采用 SMOTE[18] 增强算法使正负数据样本分布均衡化;最后构建 CNN-LSTM[19] 的深度神经网络,加以训练构建分类器,进而实现对数据竞争的检测.在实验中选取 DaCapo[20],JGF[21],IBM Contest[22],PJBench[23] 这 4 个基准测试程序套件中的 26 个基准程序进行数据竞争检测,结果表明 DeleRace 的准确率为 96.79%,与目前已有的基于深度学习的检测方法 DeepRace 相比提升了 4.65%.与 RNN 和 LSTM 相比,DeleRace 采用的 CNN-LSTM 网络也具有较高的准确率.此外,我们将 DeleRace 与已有的动态数据竞争检测工具(Said 和 RVPredict)和静态数据竞争检测工具

(SRD 和 ConRacer)进行比较,结果表明 DeleRace 可以检测出更多真实有效的数据竞争.

本文的主要贡献有 3 个方面:

1) 从 26 个不同领域的实际应用程序提取指令、方法和文件等多个级别的特征构建深度学习模型训练数据集和测试数据集.

2) 提出一种适合数据竞争检测的深度学习模型 DeleRace,使用 CNN 的卷积核提取相关特征,借助 LSTM 提取时序特征,通过 CNN 和 LSTM 的结合提升检测精度.

3) 将 DeleRace 与现有的基于深度学习的数据竞争检测工具进行了对比,并与已有的基于程序分析的数据竞争检测工具进行比较,验证了 DeleRace 的有效性.

1 基于深度学习的数据竞争检测方法

本节首先给出 DeleRace 的检测框架,然后对框架中的每个部分进行详细介绍.

1.1 检测框架

为了对数据竞争进行检测,提出了一个基于深度学习的数据竞争检测框架 DeleRace.首先,为了构建深度学习模型的训练数据集,DeleRace 从 DaCapo[20],JGF[21],IBM Contest[22],PJBench[23] 四个基准测试程序套件中选取 26 个含有数据竞争的并发程序,然后使用静态程序分析工具提取数据竞争发生位置的上下文特征信息,构造训练和测试样本,并且在样本数据中对真实有效的数据竞争进行标记.为了使提取的文本特征样本更易于被深度学习模型所处理,DeleRace 使用 Keras[24] 的嵌入层对训练样本中文本特征进行向量化.考虑到收集的并发程序中含有数据竞争正样本数可能较少,会导致正样本和负样本分布不均衡,我们使用数据增强算法增加正样本的数量,尽可能地保证正负样本均衡分布.最后,构建了一个 CNN-LSTM 深度神经网络模型,使用训练集对该模型进行训练,得到训练好的分类器,使用该分类器进行数据竞争检测.基于深度学习的数据竞争检测框架如图 1 所示:

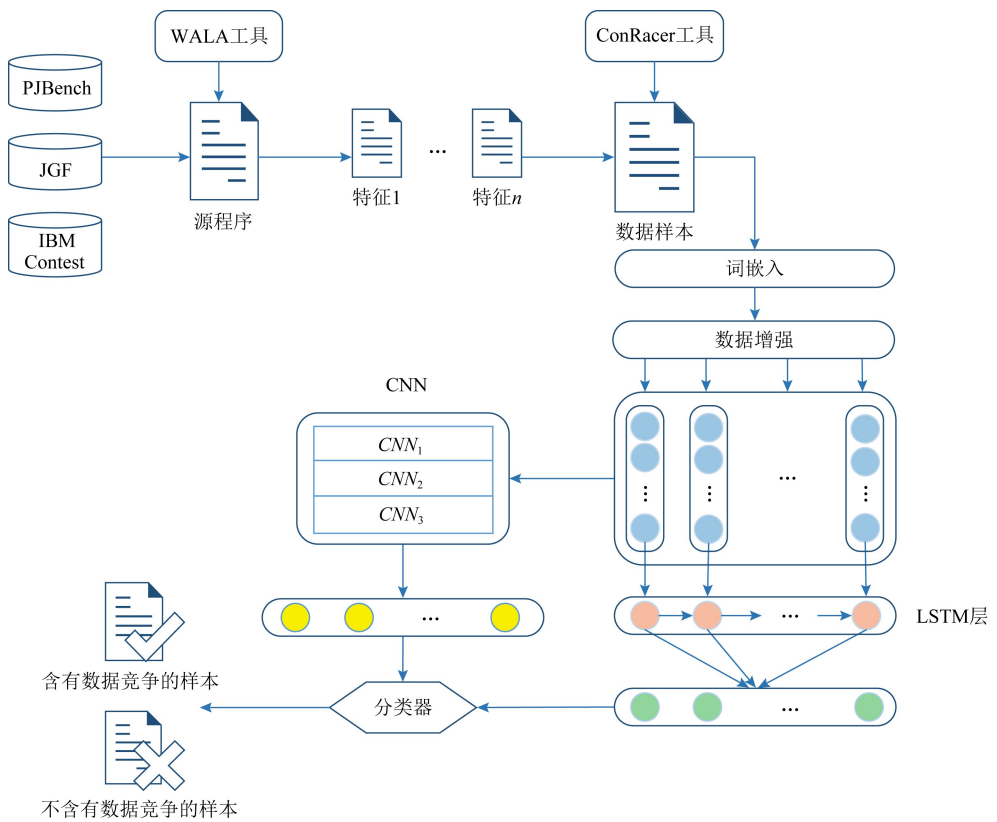

Fig. 1 The framework of DeleRace

图 1 DeleRace 方法框架

1.2 选取实际应用程序

由于目前没有公开的专门用于数据竞争检测的数据集,为了训练深度神经网络进行数据竞争检测,我们首先构建数据竞争的训练数据集.

1.3　特征提取

已有的方法在提取数据竞争特征时相对单一,无法充分体现数据竞争的产生条件,例如文献[13-14]分别提取指令和语句级别的特征,而文献[15]仅提取了文件级别的特征.为了充分提取特征,我们在构建数据集样本时充分考虑了数据竞争产生的条件,依据这些条件提取多个级别的程序相关特征.

数据竞争的产生条件包括:1)2 个或多个线程对同一个共享内存单元进行并发访问;2)至少有 1 个为写操作;3)各个操作之间没有被使用同一监视器对象的锁保护.基于这 3 个条件,我们从收集的基准程序中选取多个级别的特征来构建数据集样本,其中包括访问操作指令相关信息(如指令的 Hash 值、是否为写操作、是否被同步块包含、是否被同步方法包含)和数据竞争发生位置的相关信息(如包名、类名、方法名和变量名),其中前 4 个特征用于表明数据竞争的产生条件,后 4 个特征用于表明数据竞争发生的位置.

DeleRace 借助程序静态分析工具 WALA[16] 进行特征提取,主要操作包括:

1) 通过方法 *makeNCFABuilder*()构建程序的控制流图 *cg*.

2) 遍历控制流图 *cg*,收集所有节点 *cgNode* 下的访问操作,获取访问字段中的指令,判断该指令是否为写操作,判断是否被同步块或同步方法包含,并通过指令对应的内存地址来生成 Hash 值,将其作为变量访问的唯一标识.

3) 通过以上方法获得所有变量访问操作,将每个线程的访问操作存入集合 V 中,V 定义为

$$V=\langle isWrite, hashCode, isSyn, isSynBlock\rangle, \quad (1)$$

其中 *isWrite* 表示是否为写操作,*hashCode* 表示变量访问操作指令的 Hash 值,*isSyn* 表示是否被同步方法包含,*isSynBlock* 表示是否被同步块包含.遍历每个线程集合中的所有访问操作,与不同线程的访问操作进行对比判断,获得所有可能存在竞争的访问操作对.

4) 通过获取包名、类名、方法名以及所有静态变量和实例变量表明每对访问操作发生数据竞争的位置,判断其访问变量是否相同.

这里对特征提取后的表现形式进行演示,如表 1 和表 2 所示.表 1 展示了 IBM Contest[22] 基准测试程序套件中的 Account 程序中部分数据的数值特征信息.其中,"读写访问"列中的"1"代表写操作,"0"代表读操作;"标签"列中"1"代表构成数据竞争,"0"代表不构成数据竞争;其他列中的"1"代表是,"0"代表否,每一条数据样本包括 2 个访问操作,每个访问操作包含读写访问、Hash 值、同步方法、同步块等 4 条指令级别的特征.

Table 1　Numerical Feature

表 1　数值特征信息

样本序号	访问操作 1				访问操作 2				标签
	读写访问	Hash 值	是否被同步方法包含	是否被同步块包含	读写访问	Hash 值	是否被同步方法包含	是否被同步块包含	
1	0	1 055 461 584	0	0	0	952 975 668	0	0	0
2	0	1 055 461 584	0	0	0	1 656 402 084	0	0	0
3	1	1 578 914 784	0	0	1	1 578 914 784	0	0	1
4	1	126 596 798	0	0	1	126 596 798	0	0	1
5	1	126 596 798	0	0	1	126 596 798	0	0	1

Table 2　Text Feature

表 2　文本特征信息

样本序号	访问操作 1				访问操作 2				标签
	包名	类名	方法名	变量名	包名	类名	方法名	变量名	
1	account	Account	go	*out*	account	Account	go	*num*	0
2	account	Account	go	*out*	account	Account	go	*out*	0
3	account	Account	Service	*Bank_Total*	account	Account	Service	*Bank_Total*	1
4	account	Account	checkResult	*Bank_Total*	account	Account	checkResult	*Bank_Total*	1
5	account	Account	checkResult	*Bank_Total*	account	Account	checkResult	*Bank_Total*	1

表 2 展示了 IBM Contest[22] 基准测试程序套件中的 Account 程序中部分数据的文本特征.“标签”列中“1”表示构成数据竞争,“0”表示不构成数据竞争,每一条数据样本包括 2 个访问操作,每个访问操作均包含包名、类名、方法名、变量名等文本特征,其中包名和类名为文件级别的特征,方法名和变量名为方法级别的特征.

我们借助 ConRacer[17] 工具对真实数据竞争进行判定,并对样本数据进行标记.之所以选择 ConRacer 工具,是因为 ConRacer 在对数据竞争的分析过程中考虑了方法调用的上下文信息,误报和漏报都相对较少.然而在实际应用中 ConRacer 也不是完美的,仍存在一些误报和漏报,为了确保数据集标记的准确性,我们对标记情况进行了手动验证,保证数据集的正确性.

1.4 文本特征向量化

深度学习模型在训练时一般采用数值向量数据作为输入,通常不会直接采用文本数据.为了使训练数据和测试数据易于被深度学习模型所使用,需要把提取的文本特征转化为数值向量.在文本特征向量化的过程中,由于指令级别的特征信息本身为整数(如表 1 所示),因此无需将其向量化;而对于方法和文件级别的特征,由于在特征抽取阶段获得的均为文本数据(如表 2 所示),因此需将其转化为数值向量.

DeleRace 使用 Keras[24] 的嵌入层进行文本特征向量化,该层采用有监督的学习方式,基于已经标记好的信息进行学习并更新权重,其定义可表示为

$$f: M_i \rightarrow \boldsymbol{R}_n, \tag{2}$$

其中,M_i 表示第 i 个文本特征的整数编码;$\boldsymbol{R}_n$ 表示 M_i 对应的 n 维向量;f 是一个参数化函数映射,表示将单词映射到 n 维向量.

图 2 以 Account 测试程序中某一文件级别的特征信息为例演示了文本特征向量化的过程,这里设置词向量维度 $n=8$.首先将单词表中的单词进行词频统计并进行整数编码,将单词转换为数值向量时不区分大小写,因此单词 Account 与单词 account 的编码均为 18,其他单词 *out*,*num* 编码分别为 135,100;然后将每个单词的编码 M 经过嵌入层处理后映射为一个 8 维向量,此时得到文本向量化的表示是随机的,我们对嵌入层进行训练并更新权重;最后得到一个真正可以代表每个单词的数值向量,单词间通过各自对应的数值向量反映单词间相关性,通过计算均值使每一个单词仅用一个数值来表示.

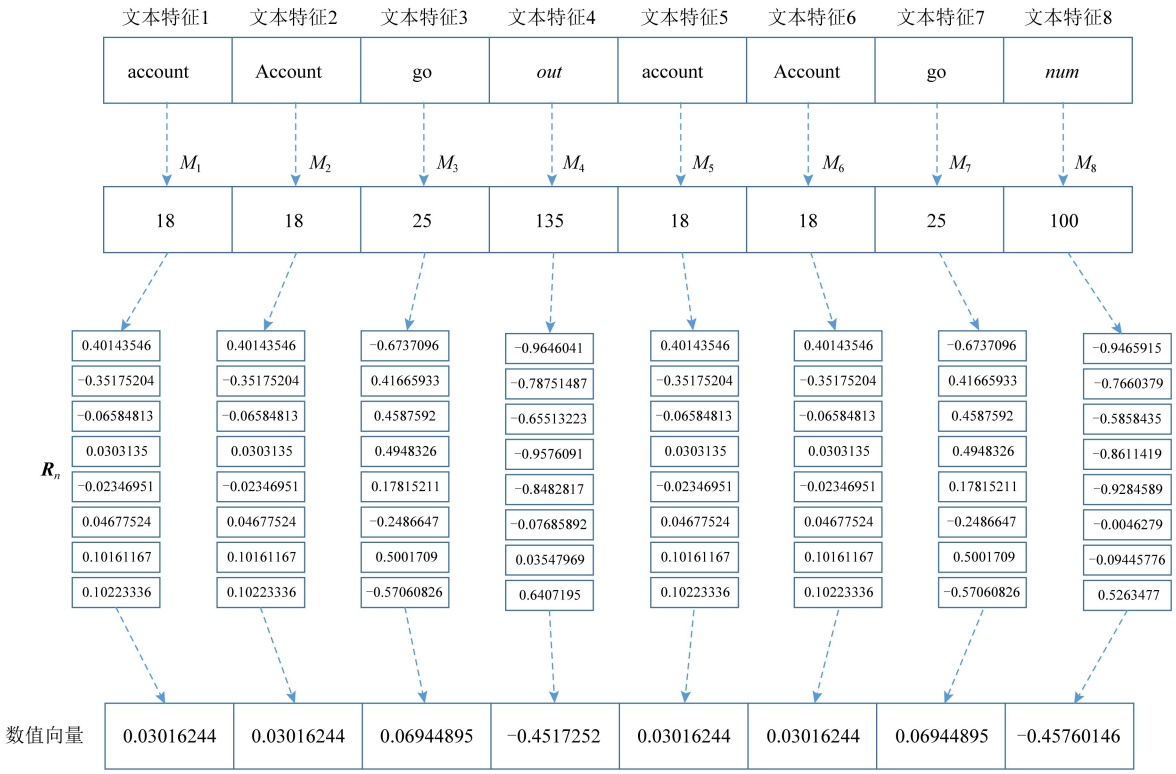

Fig. 2 Text feature vectorization

图 2 文本特征向量化

1.5　数据均衡分布

DeleRace 在样本数据的提取过程中,由于这些并发程序中数据竞争的数量很少,导致数据集中所提取的含有数据竞争的样本数远少于不含有数据竞争的样本数,这导致标签为正样本和负样本的数量极度不均衡,如果使用该数据集进行训练会严重影响深度学习模型的准确率.为了解决这个问题,通常采用欠采样(undersampling)和过采样(oversampling)2 种方法.其中,欠采样方法会从多数样本中减少训练实例,该方法只会减少不含数据竞争的样本,虽可以保证数据均衡分布,但减少了样本数量;过采样方法通过分析少数样本来增加训练实例,有助于增加数据竞争的正样本数.此外,考虑到欠采样方法一方面可能会因为减掉的数据导致某些关键信息丢失,另一方面减少训练样本也很有可能导致模型精度下降,因此本文采用过采样方法来达到数据均衡分布的目的,既可以保证特征信息的完整,又有助于扩充训练样本从而提高模型精度.

在过采样过程中,DeleRace 使用 SMOTE 算法[18]进行数据增强,它是一种合成少数类的过采样技术,其基本思想是对少数类样本进行分析,并根据少数类样本合成新样本,然后添加到数据集中,如图 3 所示.这里先选定一个正样本,找出这个正样本的 K 近邻(假设 $K=4$),随机从 K 个近邻中选择一个样本,在正样本和被选出的近邻样本的连线上随机找一个点,这个点就是我们生成的新的正样本,一直重复这个过程,直到正样本和负样本数量均衡.通过 SMOTE 算法,将原有的 12 836 条训练样本扩充到 25 438 条,从而使正样本和负样本的数据样本数量达到了均衡.

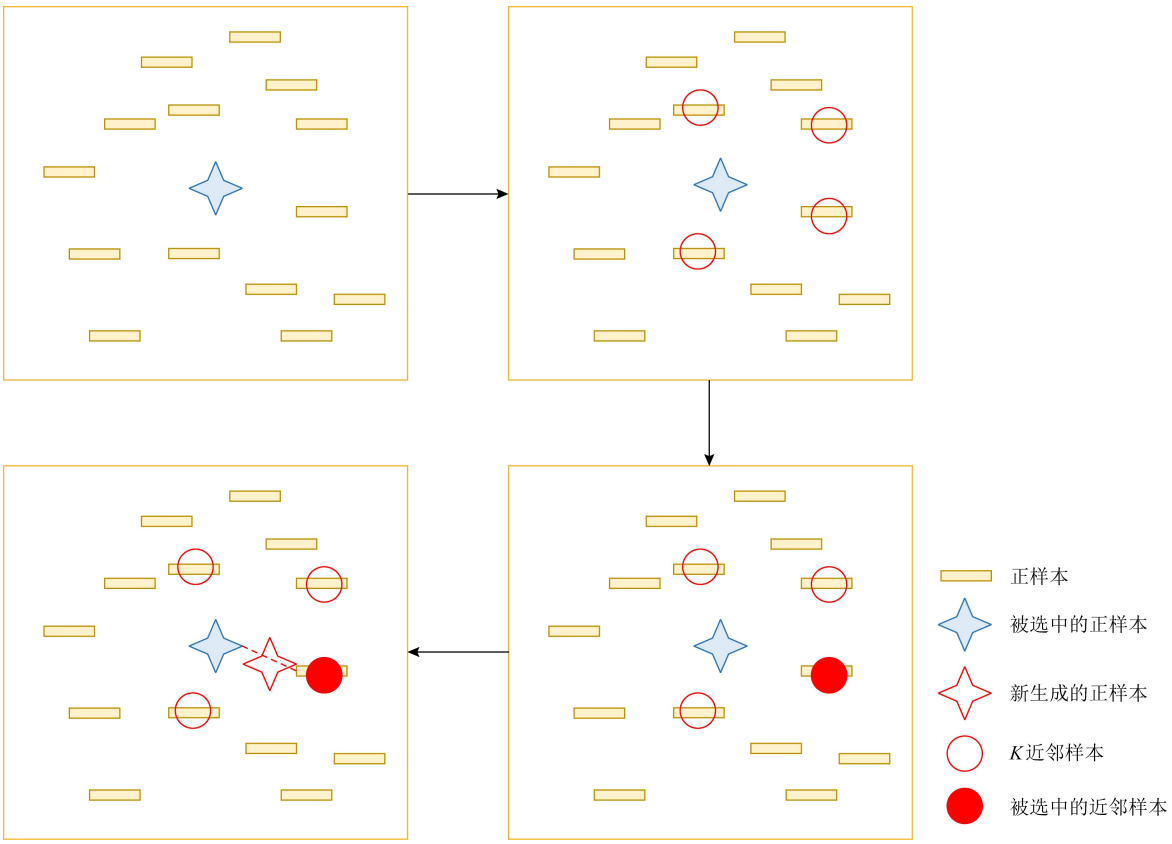

Fig. 3　SMOTE algorithm[18]

图 3　SMOTE 算法[18]

1.6　CNN-LSTM 神经网络

本文采用 CNN-LSTM 的神经网络模型,该模型在 Keras 中实现.CNN 并非只能处理图像,在 NLP 领域也能够很精准地处理分类任务,比如在情感分析[25]和观点分类[26]中都发挥了很好的作用.使用 CNN 进行文本分类最显著的优势是其无需人工手动地提取文本特征,可以自动获取基础特征并组合为高级特征,训练模型获得文本特征与目标分类之间的关系.我们借助 CNN 模型中卷积层的卷积核提取特征,然后使用最大池化层对上一层卷积层进行

降维,既降低提取特征的数据维度,又保留了具有代表性的特征.

虽然 CNN 在处理文本分类问题上存在诸多益处,但由于卷积核的存在,导致 CNN 在处理时序信号数据时存在"长期依赖"问题.针对这一问题,我们选择了 LSTM 神经网络与 CNN 进行结合.LSTM 是长短期记忆神经网络,可以有效解决"长期依赖"问题,不仅如此,LSTM 既能解决 RNN 在训练时反向传播带来的"梯度消失"问题,又能够获得源代码中的语义关系.源程序中的包名、类名、方法名和变量名多以功能命名,因此不同层次中的语义信息与数据竞争息息相关.我们选取不同层次的文本特征,通过 LSTM 层,可以提取其中的语义关系[27],例如包含关系和上下文关系等,这有助于检测数据竞争,提高检测精度.

构建的 CNN-LSTM 神经网络架构如图 4 所示:

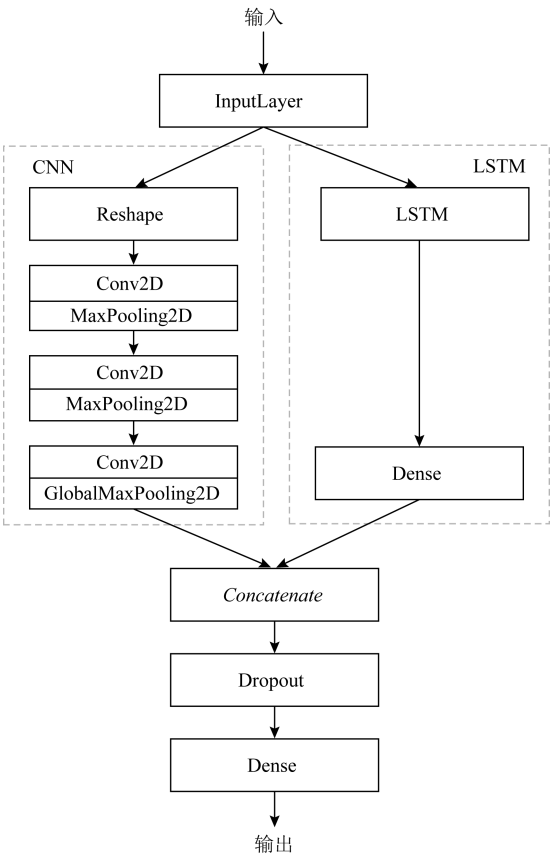

Fig. 4 CNN-LSTM deep neural network model

图 4 CNN-LSTM 深度神经网络模型

训练网络时,首先将每对访问操作的特征信息输入到 CNN-LSTM 神经网络中,DeleRace 的特征输入定义为

$$Input=\langle Number\_input, Text\_input\rangle, \quad (3)$$

$$Number\_input=\langle num_{w}, num_{h}, num_{sm}, num_{sb}\rangle,$$

$$Text\_input=\langle text_{pa}, text_{c}, text_{m}, text_{v}\rangle,$$

其中,$Input$ 表示分类器的输入,$Number\_input$ 表示数值信息的输入,num_{w},num_{h},num_{sm},num_{sb}分别表示是否为写操作、Hash 值、是否被同步方法包含和是否被同步块包含,$Text\_input$ 表示文本信息的输入,此时输入的文本信息已经被向量化,$text_{pa}$,$text_{c}$,$text_{m}$,$text_{v}$ 分别为包名、类名、方法名和变量名.由于 DeleRace 的卷积和池化都采用了 2 维操作,因此用 Reshape 函数将数值向量的 1 维矩阵转化为 2 维,然后再将输入层的输出传入 2 维卷积层中进行自动学习,每个卷积层后都有一个最大池化层来降低特征维数,避免过拟合.函数 *Concatenate* 把 CNN 输出的卷积特征和和 LSTM 提取的时序特征融合到另一个全连接层进行二分类,并通过 Dropout 方法来防止过拟合,最终输出测试程序中含有数据竞争的个数.

2 实验结果与分析

本节首先对实验配置进行介绍;然后对 DeleRace 进行了实验评估,并对结果进行了分析;最后给出了 DeleRace 与传统的数据竞争检测工具的对比.

2.1 实验配置

硬件上,所有的实验都是在 Dell Z820 工作站上进行的,该工作站配备 2 个 Intel Xeon 处理器,主频为 3.2 GHz,内存为 8 GB.软件上,操作系统使用 Windows 7 Professional,开发平台使用 Jupyter NoteBook;使用 Python3.6 和 Tensorflow1.9 作为深度学习的运行支撑环境;程序分析工具使用 WALA1.4.2,使用 Eclipse 4.5.1 作为 WALA 的运行平台,JDK 版本为 1.8.0_31.

2.2 数据集描述

表 3 列出了构建 DeleRace 训练集的基准测试程序及其配置信息,这些基准程序主要来源于 DaCapo[20],JGF[21],IBM Contest[22] 基准测试程序套件.从表 3 中可以看出,DeleRace 从 16 个训练程序中共提取了 12 836 个数据样本,由于数据集中正负样本分布不均衡,所以采用 SMOTE 算法对训练样本进行扩充,扩充后的训练样本数增长了近 1 倍,总数为 25 438 个.Lusearch 是较大型的基准测试程序,其提取的训练样本数最多,达到了 5 683 个,经过 SMOTE 算法扩充后样本数增加到 11 336 个.对于 Rax 基准测试程序,最初只提取了 23 个训练样本,

经过 SMOTE 扩充后训练样本增加到 36 个,是 16 个基准测试程序中所提训练样本最少的程序.对于其他基准测试程序,提取的训练样本数经过扩充后数量的范围在 98～4 618.

Table 3　Training Dataset

表 3　训练数据集

基准测试程序	程序功能描述	代码行数	最初训练集样本个数	扩充后训练集样本个数
Animator[22]	解释器命令集的实现算法	1 397	161	304
Crypt[21]	IDEA 加密算法	599	386	768
Elevator[22]	电梯调度算法	1 155	960	1 850
Lusearch[20]	文本搜索工具	49 785	5 683	11 336
Lufact[21]	LU 因子分解算法	806	326	648
Moldyn[21]	分子动态模拟程序	815	695	1 386
Nestedmonitors[22]	嵌套监控器算法	70	50	98
Pipline[22]	共享库	75	52	104
Rax[22]	多线程算法	52	23	36
Readerswriters[22]	读者-作家算法	285	193	382
ReplicatedCaseStudies[22]	多线程算法	890	424	844
Series[21]	傅里叶系数分析算法	476	796	1 588
SOR[21]	超松弛迭代算法	499	95	184
Sparsematmult[21]	稀疏矩阵乘法算法	497	529	1 054
TestRace[22]	简单的多线程算法	217	128	238
Tsp[22]	解决旅行商问题算法	450	2 315	4 618
总计		58 068	12 816	25 438

表 4 列出了构建 DeleRace 测试集的并发程序及其配置信息,这些测试程序主要来源于 JGF[21],IBM Contest[22],PJBench[23] 基准测试程序套件.在 Account,AirlineTickets,Boundedbuffer 等 10 个基准程序中提取数据样本作为 DeleRace 的测试集.表 4 中列出了这些测试程序及抽取的测试样本数,其中 Boundedbuffer 描述生产者-消费者算法,是所有测试程序中提取测试样本最多的程序,共提取 599 条测试样本;Critical 是模拟双线程环境的测试程序,是所有基准程序中提取测试样本最少的程序,只有 11 条测试样本;其他测试程序的测试样本数在 25～403.

Table 4　Test Dataset

表 4　测试数据集

测试程序	测试程序描述	代码行数	测试集样本个数
Account[22]	账户计数系统	87	143
AirlineTickets[22]	机票销售系统	83	139
Boundedbuffer[22]	生产者-消费者算法	334	599
Bubblesort[22]	冒泡排序算法	274	177
Bufwriter[22]	多线程缓冲池模拟程序	199	88
Critical[22]	双线程环境模拟程序	63	11
Mergesort[22]	归并排序算法	298	25
PingPong[22]	乒乓球游戏模拟程序	124	32
RayTracer[21]	光线跟踪程序	985	403
Weblech[23]	Java 实现的网站下载工具	1 464	69
总计		3 911	1 686

2.3 研究问题

在实验中,我们提出了 6 个研究问题(research question, RQ),通过回答这些问题对 DeleRace 方法进行评估.

RQ1:不同特征信息作为 DeleRace 输入信息对数据竞争检测结果有什么影响?如果只考虑几种特征输入信息,DeleRace 的性能会如何?

RQ2:DeleRace 是否能准确有效地检测出数据竞争?

RQ3:DeleRace 是否优于现有的基于深度学习的数据竞争检测工具,与其他的深度神经网络相比,DeleRace 的表现会如何?

RQ4:DeleRace 是否优于传统的基于动态分析的数据竞争检测工具?

RQ5:DeleRace 是否优于传统的基于静态分析的数据竞争检测工具?

RQ6:DeleRace 在检测数据竞争时各部分时间性能表现如何?

RQ1 关注的是不同特征信息对数据竞争检测结果的影响,通过比较部分特征信息和全部指令特征、方法特征和文件特征相结合的信息检测结果,以此来判断本文所提取的特征是否有效.

RQ2 关注的是 DeleRace 在各个测试程序中准确率、精确率、召回率和 F_1 值,以此来判断 DeleRace 是否可以准确测试出各个程序中是否含有数据竞争以及含有数据竞争的个数.

RQ3 关注的是 DeleRace 与其他深度学习算法在检测数据竞争上的性能对比.我们选择现有的基于深度学习的数据竞争检测算法 DeepRace[15] 进行对比实验,并且与 RNN 和 LSTM 神经网络进行对比.

RQ4 关注的是 DeleRace 与传统的基于动态程序分析工具在检测数据竞争上的性能对比.我们选择现有的基于动态分析的数据竞争检测算法 Said 和 RVPredict 进行对比实验.

RQ5 关注的是 DeleRace 与传统的基于静态程序分析工具在检测数据竞争上的性能对比.我们选择现有的基于静态分析的数据竞争检测算法 SRD 和 ConRacer 进行对比实验.

RQ6 关注的是 DeleRace 在检测数据竞争上的时间性能,针对测试数据集上 10 个开源程序,记录了 DeleRace 在整个检测数据竞争时的平均耗时情况.

2.4 模型评估指标

使用准确率、精确率、召回率、F_1 作为评价指标评估 DeleRace 的有效性.分类问题的混淆矩阵如表 5 所示,其中 TP 表示将正样本预测为正样本,FP 表示将负样本预测为正样本,FN 表示将正样本预测为负样本,TN 表示将负样本预测为负样本.

Table 5 Confusion Matrix of Binary Classification Problem

表 5 二分类问题的混淆矩阵

实际值	预测值		合计
	1	0	
1	TP	FN	$TP+FN$
0	FP	TN	$FP+TN$
合计	$TP+FP$	$FN+TN$	$TP+TN+FP+FN$

准确率(accuracy, ACC),表示预测正确的样本占测试集中所有样本的比例.

$$ACC=\frac{TN+TP}{TP+TN+FP+FN}. \tag{4}$$

精确率($Precision$),用于描述模型将正样本预测为正样本占测试集中实际预测为正样本的比例.

$$Precision=\frac{TP}{TP+FP}. \tag{5}$$

召回率($Recall$),用于描述模型将正样本预测为正样本占测试集中实际预测正确的比例.

$$Recall=\frac{TP}{TP+FN}. \tag{6}$$

F_1(F-Measure),用于描述精确率和召回率的加权调和平均.

$$F_1=\frac{Precision\times Recall}{Precision+Recall}. \tag{7}$$

精确率和召回率是相互影响的,通常情况下,精确率升高,召回率会随之下降,反之亦然.如果测试集中含有数据竞争的样本数量为 0,则会得到值为 1 的召回率,但精确率却会很低,因此,F_1 用来权衡精确率和召回率之间的关系,其取值范围通常在[0,1]之间,F_1 值越大表示模型性能越好.

2.5 模型参数选择

为了回答 RQ1,我们选择 5 个具有代表性的特征与本文所选择的 8 个特征进行对比,其对比结果如表 6 所示.首先研究特征提取的个数对于实验结果的影响情况,为此我们分别选择 8 个特征(见 1.3 节)和 5 个特征(包括访问指令的 Hash 值、是否含有读写操作、发生数据竞争的包名、类名以及方法名)的情况进行实验对比.实验结果如表 6 所示,其中“DeleRace-5”代表在 5 个特征下深度学习模型的实验结果,“DeleRace-8”代表 8 个特征下深度学习

模型的实验结果.从实验结果可以看出,在 DeleRace-8 情况下,无论是准确率、精确率、召回率还是 F_1 都比 DeleRace-5 情况下要高,这表明选取 8 个特征进行实验比选取 5 个特征更能提高数据竞争的检测精度.这里我们没有对更多的特征个数进行实验,主要是因为这 8 个特征是依据数据竞争的产生条件提取出来的,已经可以充分表示数据竞争的相关特征.基于该实验结果,在对 DeleRace 模型进行训练时选择特征个数为 8.

Table 6　Performance Comparison of Deep Neural Network Models with Different Feature Numbers

表 6　不同特征数量下深度神经网络模型的性能对比 %

模型	准确率	精确率	召回率	F_1
DeleRace-5	90.79	90.88	90.79	90.79
DeleRace-8	96.79	96.78	96.79	96.79

在不同特征数下 DeleRace 模型的训练过程中,我们还发现不仅特征提取的个数对分类精度有一定的影响,而且模型的迭代次数也会影响最终测试精度和训练时间开销.我们研究训练的迭代次数对于实验结果的影响情况,这主要基于 2 方面考虑:1)迭代次数过少可能会使检测精度降低;2)迭代次数过多虽然可以增加检测精度但会明显增加训练的时间开销.为了在检测精度和时间开销之间进行折中,我们通过实验确定相对合适的迭代次数,使其在提高模型检测精度的同时不会显著增加训练时间开销.图 5 给出了 DeleRace 随迭代次数的增加时训练和测试准确率的变化情况.从图 5 中可以看出,随着迭代次数的增加,准确率也会不断提高,当迭代次数增加到 50 时,准确率接近一个稳定值,即使迭代次数再增加准确率已不再有明显提升,因此在本实验中训练 DeleRace 模型的最佳迭代次数选择为 50.

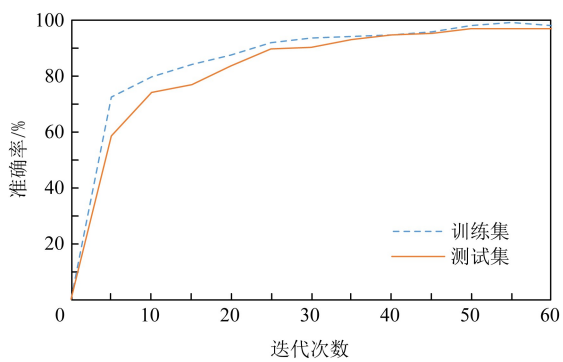

Fig. 5　The relationship between accuracy and iteration times for DeleRace

图 5　DeleRace 准确率与迭代次数的关系

2.6　DeleRace 检测结果

为了回答 RQ2,我们选用表 3 中扩充后的 25 438 个数据样本作为 DeleRace 的训练集,选用表 4 中的 10 个基准测试程序所提取出的 1 686 个数据样本作为 DeleRace 的测试集,其检测结果如表 7 所示:

Table 7　Detection Results of DeleRace

表 7　DeleRace 检测结果 %

测试程序	准确率	精确率	召回率	F_1
Account	99.30	99.44	99.30	99.34
AirlineTickets	98.56	98.49	98.56	98.02
BoundedBuffer	97.33	96.99	97.33	97.16
Bubblesort	97.18	96.95	97.18	97.06
Bufwriter	97.75	96.83	97.75	96.04
Critical	90.91	91.92	90.91	90.27
Mergesort	92.00	95.20	92.00	92.81
PingPong	93.75	95.00	93.75	93.96
RayTracer	96.04	98.31	96.00	97.16
Weblech	97.10	95.18	97.10	96.13

由表 7 可知,整体上准确率范围在 90.91%～99.30%,F_1 值在 90.27%～99.34%.Account 测试程序的准确率和 F_1 值分别为 99.30%和 99.34%,在该测试程序中,其准确率、精确率、召回率和 F_1 都是 10 个测试程序中最高的;而对于 Critical 基准测试程序,由于只提取了 11 个数据样本,其各个指标值都是 10 个基准测试中最低的,造成其性能较差的原因可能是因为含有的数据竞争数较少,进而提取的样本数也少,而深度学习模型的精度又与数据集大小有关,提供的数据集样本越多,效果越好,这可能导致 Critical 测试程序的检测精度偏低.

2.7　与其他深度神经网络方法对比

为了回答 RQ3,我们将 DeleRace 与现有的基于深度学习的数据竞争检测工具 DeepRace[15] 进行对比,并与单独使用 RNN 或 LSTM 的神经网络性能进行比较,其实验对比结果如表 8 所示:

Table 8　Performance Comparison of Different Deep Neural Network Models

表 8　不同深度神经网络模型性能对比 %

模型	准确率	精确率	召回率	F_1
RNN	88.68	90.05	88.68	88.56
LSTM	89.71	89.90	89.71	89.70
DeepRace	92.14	92.68	92.14	92.12
DeleRace	96.79	96.78	96.79	96.79

DeepRace[15]是目前已有的基于深度学习的数据竞争检测工具,由于其数据集和模型均未开源,我们无法将其直接与本文方法 DeleRace 进行比较.为了进行比较,我们根据文献[15]中介绍的方法对其工作进行了复现.在比较时,DeleRace,RNN,LSTM,DeepRace 都使用相同的训练集和测试集,遵循相同的过程和使用相同的工具来解析源代码和特征向量化,处理数据不平衡问题时也采用了相同的数据增强算法,在最大程度上保证了实验的公平性.与此同时,为了使我们的实验结果更为可靠,我们采用 10 倍交叉验证的方式来评估 DeleRace,即将训练集和数据集混合为 1 份数据集,把整个数据集随机分成 10 组,用其中 9 组作为训练集,另外 1 组作为测试集,重复这个过程,直到每组数据都作过测试集,我们取 10 次结果的平均值作为我们的最终结果.实验结果如表 8 所示.

从表 8 可以看出,DeleRace 检测的准确率为 96.79%,F_1 为 96.79%,在 4 种方法中是最高的,准确率和 F_1 分别比 DeepRace 高出约 4.65%和 4.67%,这表明本文方法 DeleRace 在检测数据竞争方面的性能优于 DeepRace.此外,我们还将 DeleRace 与 RNN 和 LSTM 网络相比,在准确率方面,DeleRace 比其他 2 种网络模型高出 7%以上,DeleRace 的 F_1 值也比其他 2 种网络模型分别高出 7.09%,8.23%.实验结果表明,无论是准确率还是 F_1,DeleRace 均优于采用 RNN 神经网络、LSTM 神经网络和 DeepRace 工具检测数据竞争的方法.

2.8 与动态数据竞争检测工具对比

为了回答 RQ4,我们将 DeleRace 与现有的基于动态程序分析的的数据竞争检测工具 Said[5]和 RVPredict[6]进行对比,其实验对比结果如表 9 所示,其中 *R-races* 表示该程序实际的数据竞争数目.

Table 9 Comparison of DeleRace and Dynamic Data Race Detection Tools

表 9 DeleRace 与动态数据竞争检测工具的对比

测试程序	*R-races*	Said			RVPredict			DeleRace		
		TP	*FN*	*FP*	*TP*	*FN*	*FP*	*TP*	*FN*	*FP*
Account	4	4	0	1	4	0	1	4	0	1
AirlineTickets	7	6	1	0	7	0	2	7	0	2
BoundedBuffer	12	10	2	0	12	0	1	12	0	4
Bubblesort	9	8	1	0	8	1	0	8	1	4
Bufwriter	2	0	2	0	2	0	0	2	0	2
Critical	8	7	1	0	8	0	0	8	0	1
Mergesort	3	3	0	4	3	0	6	3	0	2
PingPong	8	4	4	0	4	4	0	8	0	2
RayTracer	3	3	0	2	3	0	2	3	0	3
Weblech	2	1	0	1	1	0	1	2	0	1
总计	58	46	11	8	52	5	13	57	1	22

从表 9 可以看出, Said 检测到数据竞争总数为 46 个, RVPredict 检测的总数为 52 个,而 DeleRace 检测到实际竞争的个数为 57 个,明显多于其他 2 种检测工具.对于测试程序 AirlineTickets,BoundedBuffer,Bufwriter 和 Weblech,DeleRace 检测到的实际竞争个数均多于 Said 检测到的数据竞争个数;对测试程序 PingPong,DeleRace 检测到的数据竞争个数比 Said 和 RVPredict 检测到的数据竞争个数多 4 个,并且使用 DeleRace 检测大部分程序的结果与实际竞争数 ≠ *R-races* 是相同的, DeleRace 最贴近真实竞争的个数.

在漏报方面,Said 和 RVPredict 工具的漏报总数分别为 11 个和 5 个,而 DeleRace 的漏报总数为 1 个,相比较而言,DeleRace 检测数据竞争时存在较少的漏报.在这些测试程序中,只有在检测 Bubblesort 测试程序时存在 1 个漏报,对于 Said 和 RVPredict 也同样存在 1 个漏报.漏报可能的原因是将某条标签为 1 的文本特征转化成数值向量时的单词相关性较低,从而导致其发生漏报.

尽管 DeleRace 检测的真实数据竞争个数较多且漏报较少,但在误报方面 DeleRace 与其他 2 个检测工具相比还存在一定的差距.总体来说,Said 和

RVPredict 分别只有 8 个和 13 个误报,但是 DeleRace 的误报数却有 22 个.在这些程序中,误报较多的是 BoundedBuffer 和 Bubblesort 测试程序,造成误报的原因可能是在提取数据样本时标签为“0”的数据样本过多而标签为“1”的数据样本过少,造成了数据分布不均衡,尽管我们使用了 SMOTE 数据增强算法,使数据样本达到了平衡,但还是不可避免地造成了一些误报.DeleRace 在误报方面的不足,将驱使我们进一步完善该方法.

2.9　与静态数据竞争检测工具对比

为了回答 RQ5,我们将 DeleRace 与现有的基于静态程序分析的的数据竞争检测工具 SRD[5] 和 ConRacer[17] 进行对比,其实验对比结果如表 10 所示.

从表 10 可以看出,DeleRace 检测到的数据竞争总数比 SRD 检测到的个数多 17 个,漏报个数比 SRD 少 15 个,在误报方面虽然与 SRD 相等,但总体而言,DeleRace 的性能是优于静态数据竞争工具 SRD 的.

Table 10　Comparison of DeleRace and Static Data Race Detection Tools

表 10　DeleRace 与静态数据竞争检测工具的对比

测试程序	*R-races*	SRD			ConRacer			DeleRace		
		TP	*FN*	*FP*	*TP*	*FN*	*FP*	*TP*	*FN*	*FP*
Account	4	4	0	1	4	0	1	4	0	1
AirlineTickets	7	7	0	3	7	0	2	7	0	2
BoundedBuffer	12	12	0	4	12	0	0	12	0	4
Bubblesort	9	1	8	0	8	1	0	8	1	4
Bufwriter	2	2	0	0	2	0	0	2	0	2
Critical	8	6	2	0	8	0	1	8	0	1
Mergesort	3	3	0	12	1	2	0	3	0	2
PingPong	8	4	4	0	8	0	2	8	0	2
RayTracer	3	1	2	0	1	2	0	3	0	3
Weblech	2	0	0	2	2	0	0	2	0	1
总计	58	40	16	22	53	5	6	57	1	22

我们还将 DeleRace 与 ConRacer 进行了对比.对于检测到的实际竞争个数,DeleRace 所检测到的个数为 57 个,而 ConRacer 是 53 个,本文方法 DeleRace 比 ConRacer 所检测到的个数多 4 个.漏报方面,本文方法 DeleRace 比 ConRacer 的漏报个数少 4 个.尽管 DeleRace 检测的真实数据竞争个数较多且漏报较少,但在误报方面 DeleRace 与 ConRacer 相比依旧存在一些差距,ConRacer 的误报个数仅为 6 个,但本文方法 DeleRace 却为 22 个,造成误报较多的原因在 2.8 节已经进行了说明.我们计划在下一步的工作中进行改进.

虽然 ConRacer 工具检测到的实际竞争个数比本文方法 DeleRace 少,且漏报个数比 DeleRace 多,但与本文所提到的 Said,RVPredict 和 SRD 这 3 种基于程序分析的数据竞争检测方法相比,ConRacer 不仅检测到的实际竞争个数最多,而且误报和漏报均最少,因此 ConRacer 依旧是较好的检测工具.针对判定时所出现的误报和漏报情况,我们对标记情况进行了手动验证,以保证数据集的正确性.

2.10　时间性能评估

对于回答 RQ6,我们评估了 DeleRace 在检测数据竞争时的时间性能.表 11 记录了 DeleRace 检测数据竞争的整体耗时情况,总时长为 3 657.14 s,其中耗时最久的步骤是深度神经网络模型的训练时间,共耗时 3 492.42 s,占据整体检测时长的 95%,主要原因是神经网络的时间复杂度和空间复杂度都会对模型的训练时间产生影响,如果复杂度过高,很容易导致模型在训练过程中消耗过多的时间,而 DeleRace 采用了 3 个卷积层和 1 个 LSTM 层处理数据,获得

Table 11　The Time of Each Step in DeleRace

表 11　DeleRace 完成各个步骤所花费的时间

步骤	花费时间/s
特征提取	86.17
嵌入层训练	74.81
模型训练	3 492.42
模型测试	3.74
总计	3 657.14

了不同的特征并加以合并,层数深且操作过程较为复杂,并且迭代次数过多也会使得训练时间过长,因此在训练模型过程中花费时间较多,耗时比例较高.

2.11 有效性威胁

本节对实验过程中威胁有效性的4个因素进行了讨论.

1) 本文仅选择来自DaCapo,JGF,IBM Contest,PJBench这4个基准测试组件中的并发程序,从这些程序中提取的数据竞争数据集并不能代表所有程序,因为不同程序可能展现不同的数据样本.为了缓解这个有效性威胁,我们选择了26个测试程序,这些程序分别来自不同领域,尽可能保证数据集来源的多样性.

2) 我们在对训练数据集进行标记时采用了ConRacer工具,尽管该工具采用了上下文敏感的程序分析方法,可以有效地报告数据竞争,但该工具仍存在误报和漏报的情况.为了解决这一问题,我们对报告的数据竞争采用手动的方式检查数据竞争位置的代码,尽可能地排除误报和漏报,最大程度上保证了数据集的准确性.

3) 将文本特征转化为数值向量时,转化相似度的高低会影响最终结果的精度.本文采用Keras中的嵌入层进行文本向量化,通过对Keras的嵌入层进行训练并对其参数进行调节使其转化准确率可达98.7%,虽然没有达到100%的准确率,但通过该技术转化的数值向量已经在很大程度上接近数据竞争的文本特征,有效减少了文本向量化对最终结果的影响.

4) 在数据增强时我们选择的算法会对训练的最终效果有一定的影响.本文选择的SMOTE算法是基于随机过采样算法的一种改进方案,由于随机过采样采取简单复制样本的策略来增加少数类样本,容易使模型过拟合,而SMOTE算法采用KNN技术生成新样本[28-29],通过计算每个少数类样本的K个近邻,并从中随机挑选N个样本进行随机线性插值,最终构造新的少数类样本.因此使用SMOTE算法增加正样本个数,在扩充训练集时有效减小数据增强对模型训练的影响.

3 相关工作

数据竞争检测的相关研究仍是目前研究的热点内容之一,所采用的方法有很多,目前可分为基于程序分析的数据竞争检测方法以及基于机器学习和深度学习的数据竞争检测方法.基于程序分析的数据竞争检测方法通常又分为动态检测、静态检测和动静结合的检测方法.

动态检测通过运行源程序时插桩等方法获取并记录程序运行时状态.Said等人[5]提出了一种基于SMT解算器的符号分析方法,可以有效地分析线程调度,准确定位数据竞争.RVPredict[6]将数据竞争检测作为约束求解问题,利用可满足性模理论(satisfiability modulo theories, SMT)求解器查找数据竞争.SlimFast[7]通过减少数据冗余、内存使用和运行时间来检测数据竞争并提高检测效率.

静态方法是基于静态源代码分析,通过程序验证或符号执行的方式分析源码语义或者程序控制流.RELAY[8]是一种基于流敏感和过程间分析的静态数据竞争检测工具.Elmas[9]是基于模型检测理论提出的一种检测方法,通过对程序中锁操作路径进行分析并通过发生序关系过滤结果.SRD[10]采用程序切片技术静态判断访问事件之间的发生序关系并结合别名分析等静态分析技术检测数据竞争.

动静结合的检测方法是先静态找出所有可能的数据竞争,再利用动态分析检测程序.RaceTracker[11]采用动静结合的检测方法,首先使用当前的静态检测器来产生潜在的竞争,然后潜在竞争的代码位置进行插桩来识别数据竞争.AsampleLock[12]是基于优化的FastTrack[30]算法和锁模式的动态混合数据竞争检测算法,利用采样技术监控同一时刻同时运行的并发线程函数对,再通过预竞争检测获得真正的数据竞争的内存访问对.

有些研究人员开始使用机器学习和深度学习方法来检测数据竞争.Tehrani等人[15]提出基于深度学习的数据竞争检测工具DeepRace,首先通过变异分析生成特定的数据竞争类型,再通过为每个源文件生成AST构造数据集,最后将向量化的数值输入到CNN中进行训练,其检测准确率在83%~86%.孙家泽等人提出了AIRaceTest[13]和ADR[14]检测数据竞争工具,AIRaceTest是基于随机森林的数据竞争指令级的检测工具,首先基于HB关系和Lockset算法指令级检测数据竞争,并利用其分析结果训练数据竞争随机森林检测模型,模型精度为92%.ADR是基于Adaboost模型的数据竞争语句级检测工具,将插桩得到的指令内存信息进行语句级转化,提取出相关特征后构建Adaboost数据竞争检测模型.与AIRaceTest和DeepRace相比,DeleRace通过提取多个特征,模型性能更好,准确率也更高.

4 总　　结

本文提出一种基于深度学习的数据竞争检测方法,该方法首先利用 WALA 工具从多个实际应用程序中提取指令、方法和文件等级别的多个特征,对其向量化并构造训练样本数据,通过 ConRacer 工具对真实数据竞争进行判定进而标记样本数据,采用 SMOTE 增强算法使数据样本均衡化,最后构建并训练 CNN-LSTM 深度神经网络实现对数据竞争的检测.在实验中选取 10 个基准测试程序验证了该方法的有效性,结果表明 DeleRace 的数据竞争检测准确率为 96.79%,高于目前已有的基于深度学习的检测方法.此外,我们将 DeleRace 与已有的动态数据竞争检测工具(Said 和 RVPredict)和静态数据竞争检测工具(SRD 和 ConRacer)进行比较,验证了本文方法的有效性.

进一步的研究工作包括:1)针对本文方法误报较多的问题,将在未来的工作中通过扩大训练数据集、增加含有数据竞争的标签样本和采用更多软件分析等方法减少误报,并且选取更多的基准测试程序对模型进行训练,提高工具的普适性;2)本文使用 CNN-LSTM 的神经网络来对数据竞争进行检测,虽然准确率可达 96.79%,但仍有可提升的空间,我们将继续尝试对深度学习模型进行优化,进一步提高检测精度.

作者贡献声明:张杨负责论文想法的提出、方法设计、实验指导、数据整理与分析、论文的写作与修改;乔柳进行实验设计与探究、深度学习模型实现、实验数据整理与分析、论文的写作与修改;东春浩负责深度学习模型实现、部分实验数据整理、论文的修改;高鸿斌进行实验指导、论文的修改.

参　考　文　献

[1] Yu Zhen, Yang Zhen, Su Xiaohong, et al. Research review on multithread program data race detection and verification methods [J]. Intelligent Computer and Application, 2017, 7(3): 123-126 (in Chinese)
(禹振, 杨振, 苏小红, 等. 多线程程序数据竞争检测和验证方法研究综述[J]. 智能计算机与应用, 2017, 7(3): 123-126)

[2] Gu Yizi, Mellor-Crummey J. Dynamic data race detection for OpenMP programs [C] //Proc of the 26 Int Conf for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2018: 767-778

[3] Chen Jun, Zhou Kuanjiu, Jia Min. Data race static detection method for multi-threaded parallel programs [J]. Computer Engineering and Design, 2017, 38(5): 1264-1272 (in Chinese)
(陈俊, 周宽久, 贾敏. 多线程并行程序数据竞争静态检测方法[J]. 计算机工程与设计, 2017, 38(5): 1264-1272)

[4] Sen A, Kalaci O. Hybrid data race detection for multicore software [J]. Computing and Informatics, 2018, 37(1): 186-212

[5] Said M, Wang Chao, Yang Zijiang, et al. Generating data race witnesses by an SMT-based analysis [C] //Proc of Formal Methods Symp. Berlin: Springer, 2011: 313-327

[6] Huang J, Meredith P O, Rosu G. Maximal sound predictive race detection with control flow abstraction [J]. ACM SIGPLAN Notices, 2014, 49(6): 337-348

[7] Peng Yuanfeng, DeLozier C, Eizenberg A, et al. SlimFast: Reducing metadata redundancy in sound and complete dynamic data race detection [C] //Proc of the 10th Int Parallel and Distributed Processing Symp (IPDPS). Los Alamitos, CA: IEEE Computer Society, 2018: 835-844

[8] Voung J W, Jhala R, Lerner S. RELAY: Static race detection on millions of lines of code [C] //Proc of the 6th Joint Meeting of the European Software Engineering Conf and the ACM SIGSOFT Int Symp on Foundations of Software Engineering. New York: ACM, 2007: 205-214

[9] Taft S T, Schanda F, Moy Y. High-integrity multitasking in spark: Static detection of data races and locking cycles [C] //Proc of the 17th Int Symp on High Assurance Systems Engineering (HASE). Piscataway, NJ: IEEE, 2016: 238-239

[10] Zhang Yang, Liang Yanan, Zhang Dongweng, et al. Data race detection approach in concurrent programs [J]. Journal of Computer Applications, 2019, 39(1): 67-71

[11] Zhen Yang, Zhen Yu, Su Xiaohong, et al. RaceTracker: Effective and efficient detection of data races [C] //Proc of the 20th ACM Symp on Operating Systems Principles. Los Alamitos, CA: IEEE Computer Society, 2016: 293-300

[12] Li Mengke, Zheng Qiusheng, Wang Lei. Dynamic hybrid data race detection algorithm based on sampling technology [J]. Computer Science, 2020, 47(10): 315-321 (in Chinese)
(李梦珂, 郑秋生, 王磊. 基于采样技术的动态混合数据竞争检测算法[J]. 计算机科学, 2020, 47(10): 315-321)

[13] Sun Jiaze, Yang Jiawei, Yang Zijiang. Multithreaded program data race random forest instruction level detection model [J]. Journal of Tsinghua University: Natural Science Edition, 2020, 60(10): 804-813 (in Chinese)
(孙家泽, 阳伽伟, 杨子江. 多线程程序数据竞争随机森林指令级检测模型[J]. 清华大学学报: 自然科学版, 2020, 60(10): 804-813)

[14] Sun Jiaze, Yi Gang, Shu Xinfeng. Data race statement level detection in concurrent programs based on Adaboost model [J]. Computer Engineering, 2021, 47(12): 215-220 (in Chinese)

(孙家泽, 易刚, 舒新峰. 基于 Adaboost 模型的并发程序数据竞争语句级检测[J]. 计算机工程, 2021, 47(12): 215-220)

[15] Tehrani A, Khaleel M, Akbari R, et al. DeepRace: Finding data race bugs via deep learning[J]. arXiv, preprint, arXiv: 1907.07110, 2019

[16] IBM. Watson libraries for analysis (WALA)[EB/OL]. [2021-04-25]. http://wala.sourceforge.net

[17] Zhang Yang, Liu Huan, Qiao Liu. Context-sensitive data race detection for concurrent programs [J]. IEEE Access, 2021, 9: 20861-20867

[18] Chawla N V, Bowyer K W, Hall L O, et al. SMOTE: Synthetic minority over-sampling technique [J]. Journal of Artificial Intelligence Research, 2002, 16: 321-357

[19] Li Songzhou, Xie Gang, Ren Jinchang, et al. Urban PM2.5 concentration prediction via attention-based CNN-LSTM [J]. Applied Sciences, 2020, 10(6): 1953-1970

[20] Blackburn S M, Garner R, Hoffmann C, et al. The DaCapo benchmarks: Java benchmarking development and analysis [C] //Proc of the 21st Annual ACM SIGPLAN Conf on Object-Oriented Programming Systems, Languages, and Applications. New York: ACM, 2006: 169-190

[21] Smith L A, Bull J M, Obdrizalek J. A parallel Java grande benchmark suite [C] //Proc of the 2001 ACM/IEEE Conf on Supercomputing. Piscataway, NJ: IEEE, 2001: 6-6

[22] Farchi E, Nir Y, Ur S. Concurrent bug patterns and how to test them [C] /Proc of the 1st Int Parallel and Distributed Processing Symp. Los Alamitos, CA: IEEE Computer Society, 2003: 286-296

[23] Yu Misun, Bae D H. SimpleLock: Fast and accurate hybrid data race detection [J]. The Computer Journal, 2016, 59 (6): 793-809

[24] François C. The Python deep learning library (Keras) [EB/OL]. [2021-04-27]. https://ke-ras.io

[25] Chen Yahui. Convolutional neural network for sentence classification [D]. Waterloo: University of Waterloo, 2015

[26] Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences [J]. arXiv, preprint, arXiv:1404.2188, 2014

[27] Guo Xueliang, Shi Chongyang, Jiang He. Deep semantic-based feature envy identification [C] //Pro of the 11th Asia-Pacific Symp on Internetware. New York: ACM, 2019 [2021-04-26]. https://doi.org/10.1145/3361242. 3361257

[28] Mani I, Zhang Jianping. KNN approach to unbalanced data distributions: A case study involving information extraction [C] //Proc of Workshop on Learning from Imbalanced Datasets. San Diego, CA: ICML, 2003: 126-130

[29] Zhang Yang, Dong Chunhao. MARS: Detecting brain class/method code smell based on metric-attention mechanism and residual network [J/OL]. Journal of Software: Evolution and Process, 2021, E2403: 1-15 [2022-03-22]. https://doi.org/10.1002/smr.2403

[30] Flanagan C, Freund S N. FastTrack: Efficient and precise dynamic race detection [J]. ACM SIGPLAN Notices, 2009, 44(6): 121-133

Zhang Yang, born in 1980. PhD, associate professor. Senior member of CCF. His main research interests include concurrent defects, deep learning and software refactoring for parallelism.

张 杨, 1980 年生. 博士, 副教授, CCF 高级会员. 主要研究方向为并发缺陷、深度学习、软件重构.

Qiao Liu, born in 1996. Master candidate. Her main research interests include deep learning and software refactoring for parallelism.

乔 柳, 1996 年生. 硕士研究生. 主要研究方向为深度学习、软件重构.

Dong Chunhao, born in 1996. Master candidate. His main research interests include deep learning and software refactoring for parallelism.

东春浩, 1996 年生. 硕士研究生. 主要研究方向为深度学习、软件重构.

Gao Hongbin, born in 1964. Master, professor. His main research interests include deep learning and software refactoring for parallelism.

高洪斌, 1964 年生. 硕士, 教授. 主要研究方向为深度学习、软件重构.