软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(7):3071−3092 [doi: 10.13328/j.cnki.jos.007110]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

神经程序修复领域数据泄露问题的实证研究*

李卿源, 钟文康, 李传艺, 葛季栋, 骆 斌

(计算机软件新技术国家重点实验室(南京大学), 江苏 南京 210023)

通信作者: 葛季栋, E-mail: gjd@nju.edu.cn

摘 要: 修复软件缺陷是软件工程领域一个无法回避的重要问题, 而程序自动修复技术则旨在自动、准确且高效地修复存在缺陷的程序, 以缓解软件缺陷所带来的问题. 近年来, 随着深度学习的快速发展, 程序自动修复领域兴起了一种使用深度神经网络去自动捕捉缺陷程序及其补丁之间关系的方法, 被称为神经程序修复. 从在基准测试上被正确修复的缺陷的数量上看, 神经程序修复工具的修复性能已经显著超过了非学习的程序自动修复工具. 然而, 近期有研究发现: 神经程序修复系统性能的提升可能得益于测试数据在训练数据中存在, 即数据泄露. 受此启发, 为了进一步探究神经程序修复系统数据泄露的原因及影响, 更公平地评估现有的系统: (1) 对现有神经程序修复系统进行了系统的分类和总结, 根据分类结果定义了神经程序修复系统的数据泄露, 并为每个类别的系统设计了数据泄露的检测方法; (2) 依照上一步骤中的数据泄露检测方法对现有模型展开了大规模检测, 并探究了数据泄露对模型真实性能与评估性能间差异的影响以及对模型本身的影响; (3) 分析现有神经程序修复系统数据集的收集和过滤策略, 加以改进和补充, 在现有流行的数据集上, 基于改进后的策略构建了一个纯净的大规模程序修复训练数据集, 并验证了该数据集避免数据泄露的有效性. 由实验结果发现: 调研的 10 个神经程序修复系统在基准测试集上均出现了数据泄露, 其中, 神经程序修复系统 RewardRepair 的数据泄露问题较为严重, 在基准测试集 Defects4J (v1.2.0)上的数据泄露达 24 处, 泄露比例高达 53.33%. 此外, 数据泄露对神经程序修复系统的鲁棒性也造成了影响, 调研的 5 个神经程序修复系统均因数据泄露产生了鲁棒性降低的问题. 由此可见, 数据泄露是一个十分常见的问题, 且会使神经程序修复系统得到不公平的性能评估结果以及影响系统在基准测试集上的鲁棒性. 研究人员在训练神经程序修复模型时, 应尽可能避免出现数据泄露, 且要考虑数据泄露问题对神经程序修复系统性能评估产生的影响, 尽可能更公平地评估系统.

关键词: 程序自动修复; 神经程序修复; 深度学习; 数据泄露; 程序修复数据集

中图法分类号: TP311

中文引用格式: 李卿源, 钟文康, 李传艺, 葛季栋, 骆斌. 神经程序修复领域数据泄露问题的实证研究. 软件学报, 2024, 35(7): 3071−3092. http://www.jos.org.cn/1000-9825/7110.htm

英文引用格式: Li QY, Zhong WK, Li CY, Ge JD, Luo B. Empirical Study on Data Leakage Problem in Neural Program Repair. Ruan Jian Xue Bao/Journal of Software, 2024, 35(7): 3071−3092 (in Chinese). http://www.jos.org.cn/1000-9825/7110.htm

Empirical Study on Data Leakage Problem in Neural Program Repair

LI Qing-Yuan, ZHONG Wen-Kang, LI Chuan-Yi, GE Ji-Dong, LUO Bin

(National Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

Abstract: Repairing software defects is an inevitable and significant problem in the field of software engineering, while automated

* 基金项目: 国家重点研发计划(2022YFF0711404); 江苏省第六期“333 工程”领军型人才团队项目; 江苏省自然科学基金(BK20201250)

本文由“面向复杂软件的缺陷检测与修复技术”专题特约编辑张路教授、刘辉教授、姜佳君副研究员、王博博士推荐.

收稿时间: 2023-09-11; 修改时间: 2023-10-30; 采用时间: 2023-12-14; jos 在线出版时间: 2024-01-05

CNKI 网络首发时间: 2024-03-13

program repair (APR) techniques aim to alleviate software defect problem by repairing the defective programs automatically, accurately, and efficiently. In recent years, with the rapid development of deep learning, the field of automated program repair has emerged a method that utilizes deep neural networks to automatically capture the relationship between defective programs and their patches, called neural program repair (NPR). In terms of the number of defects that can be correctly repaired on the benchmark, NPR tools have significantly outperformed non-deep learning APR tools. However, a recent study found that the performance improvement of NPR systems may be due to the presence of test data in the training data, i.e., the data leakage. Inspired by this, to further investigate the causes and effects of data leakage in NPR systems and to evaluate existing systems more fairly, this study: (1) systematically categorizes and summarizes the existing NPR systems, defines the data leakage of NPR systems based on this classification, and designs the data leakage detection method for each category of system; (2) conducts a large-scale testing of existing models according to the data leakage detection method in the previous step and investigates the effect of data leakage on model realism and evaluation performance and the impact on the model itself; (3) analyzes the collection and filtering strategies of existing NPR system datasets, improves and supplements them, then constructs a pure large-scale NPR training dataset based on the improved strategy with the existing popular dataset, and verifies the effectiveness of this dataset in preventing data leakage. From the experimental results, it is found that the ten NPR systems studied in this investigation all had data leakage on the evaluation dataset, among which the NPR system RewardRepair had the more serious data leakage problem, with 24 data leaks on the Defects4J (v1.2.0) benchmark, and the leakage ratio was as high as 53.33%. In addition, data leakage has an impact on the robustness of the NPR system, and all five NPR systems investigated had reduced robustness due to data leakage. As a result, data leakage is a very common problem and can lead to unfair performance evaluation results of NPR systems and affect therobustness of the NPR system on the benchmark. When training NPR models, researchers should avoid data leakage as much as possible and consider the impact of data leakage on the evaluation of the performance of NPR systems to evaluate the NPR systems as fairly as possible.

Key words: automated program repair (APR); neural program repair; deep learning; data leakage; program repair dataset

程序自动修复(automated program repair, APR)[1,2]技术通过自动生成补丁, 能够帮助开发人员修复缺陷程序, 以保障软件质量, 因此成为软件工程领域一个备受关注的研究方向. 近年来, 随着深度学习(deep learning, DL)的兴起, 一种基于学习(learning-based)的程序自动修复技术开始受到研究人员的重视, 被称为神经程序修复(neural programrepair, NPR)[3]. 非学习的程序自动修复技术大多依赖于程序分析, 如基于启发式搜索(例如 GenProg[4]、ASTOR[5]、ARJA[6]和 SimFix[7])、基于语义约束(例如 DynaMoth[8]、Nopol[9]、Astor[10]和 Angelix[11])以及基于修复模板(例如 AVATAR[12]、TBar[13]、PAR[14]和 SketchFix[15])的修复技术. 相比之下, 神经程序修复的修复效果已经远远超过了这些非学习的程序自动修复[16], 并且神经程序修复能够实现端到端的工作流程, 使得修复过程更加自动化.

然而有研究表明: 数据泄露会导致对神经网络的评估结果膨胀, 进而导致对其性能评估的结果不真实[17]. 我们认为: 数据泄露同样会使神经程序修复系统在基准测试集上的修复效果膨胀; 这些出现数据泄露的缺陷被修复的真正原因可能是模型在训练时, 已经学习到了该缺陷的修复信息. 此外, 还有研究表明: 数据泄露会对神经网络的泛化性造成影响, 且会造成神经网络出现过拟合的现象. 但目前, 对于神经程序修复系统中数据泄露的研究普遍存在着一些问题: 首先, 由于神经程序修复系统的复杂性, 导致还没有一个对神经程序修复系统中数据泄露的明确定义; 其次, 现有研究大多只是在基准测试上进行简单的字符串匹配检测, 并未形成系统的数据泄露检测方法; 最后, 还没有研究进一步分析数据泄露对神经程序修复系统所造成影响. 因此, 亟需一项系统的实证研究, 对神经程序修复系统中的数据泄露进行科学的定义, 并对应产生一套合理的检测手段, 以及进一步分析数据泄露对神经程序修复系统所产生的影响. 因此, 我们在现有研究的基础上, 对神经程序修复系统中的数据泄露问题进行了一次系统的实证研究, 探究数据泄露在神经程序修复系统中的严重程度、数据泄露对现有神经程序修复系统的性能评估造成的影响以及数据泄露对模型本身造成的影响. 本文的主要贡献如下.

(1) 定义了神经程序修复系统中的数据泄露, 对应不同类型的数据泄露设计了相应的检测方法, 将检测方法合并成为一个神经程序修复系统数据泄露检测工具 NPRLeakageFinder, 并使用该工具对神经程序修复系统进行数据泄露检测;

(2) 探究了数据泄露对现有的神经程序修复系统的性能评估造成的影响以及对其鲁棒性造成的影响;

(3) 针对数据泄露问题提出解决方法, 设计了一套神经程序修复系统数据集的收集、过滤和划分策略,

参照该策略构建了一个纯净的数据集 Clean4J_Benchmark 供神经程序修复系统使用, 评估了该数据集的效用, 同时验证了所提出的数据集构建策略的有效性.

本文第 1 节介绍与神经程序修复系统相关的背景知识. 第 2 节介绍神经程序修复系统中数据泄露问题的相关研究. 第 3 节介绍本文的研究计划, 包括定义研究问题、设计实验方法等. 第 4 节归纳研究结果, 并得出研究结论. 第 5 节分析可能对实证研究的有效性产生影响的因素. 第 6 节对神经程序修复系统中的数据泄露问题进行总结与展望.

1 背景知识

1.1 神经程序修复

神经程序修复是一种狭义上的补丁生成技术, 神经程序修复系统利用各种深度神经网络(deep neural network, DNN)进行缺陷代码到正确代码的变换. 该系统通常采用序列到序列(sequence to sequence, Seq2Seq)架构的深度神经网络进行缺陷修复. Seq2Seq 架构的模型通常被用于自然语言处理(natural language processing, NLP)领域的神经机器翻译任务, 而基于 Seq2Seq 架构的神经程序修复模型, 修复缺陷的思想本质上与翻译任务类似[18,19], 是利用神经网络实现从缺陷代码的语义空间向补丁代码的语义空间的转换, 在一定程度上可以理解为将缺陷代码“翻译”成正确代码. 在训练过程中, 神经程序修复系统的目标是尽可能拟合真实(ground-truth)补丁每一个词元(token)的概率分布. 如图 1 所示, 神经程序修复系统生成一个补丁一般可分为 5 个阶段.

(1) 预处理(preprocessing): 预处理阶段将给定的缺陷代码转变成神经网络可接受的形式, 预处理经常使用到自然语言处理中的分词(tokenization)和嵌入(embedding)等技术.

(2) 编码(encoding): 编码阶段将预处理阶段生成的嵌入向量进行进一步编码, 这一步是为了让神经网络理解缺陷代码的深层语义.

(3) 解码(decoding): 解码阶段将编码得到的上下文向量进行解码, 生成预测补丁当前时间步词元的概率分布.

(4) 补丁重排序(re-ranking): 重排序阶段将模型生成的补丁进行重新排序, 将更可能是正确修复的补丁前置.

(5) 补丁验证(validation): 验证阶段将重排序后的补丁在测试上进行验证, 得到似真补丁. 最后, 对于程序自动修复工具, 一般还要对似真补丁进行人工校对(manualcheck), 检查是否能够真正修复对应的缺陷.

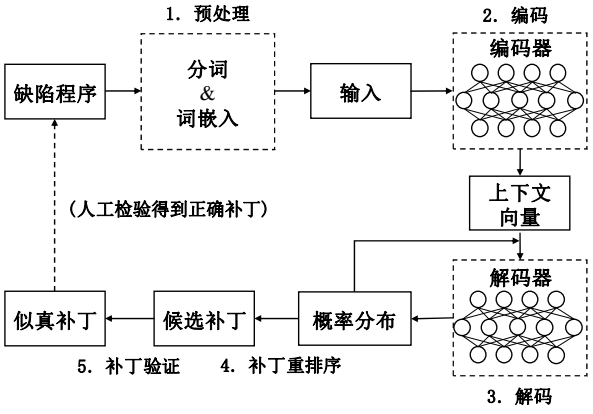

图 1 神经程序修复系统的一般流程图

近年来, 随着大语言模型(large language model, LLM)的蓬勃发展, 利用大语言模型微调(fine tuning)或零样本学习(zero-shot learning)的神经程序修复系统(例如 InferFix[20]、FitRepair[21]、AlphaRepair[22]和 SELF-

DEBUGGING[23])展示出更为强大的缺陷修复能力[24]. 这一方面得益于模型的训练数据和参数量都十分庞大, 另一方面得益于在修复缺陷程序时添加了一些提示(prompt)[25,26], 让模型知道当前要进行的是程序修复任务.

1.2 神经程序修复系统中的数据泄露

在深度学习领域, 数据泄露是一种常见的问题, 但由于神经程序修复模型具有多样的训练方式和多样的输入输出方式, 导致对其数据泄露的判定较为模糊. 因此, 本文对神经程序修复系统中的数据泄露设计了以下分类的定义.

- 缺陷-补丁对泄露(bug-fix pair). 该类型的数据泄露是指: 神经程序修复系统在基准测试集上的某条可修复的缺陷-补丁对出现在了训练数据当中, 即该条可修复的缺陷-补丁对中的缺陷部分出现在了某条训练数据的缺陷代码中, 同时, 神经程序修复系统生成的补丁出现在了同一条训练数据的补丁代码中.
- 缺陷代码泄露(buggy code). 该类型的数据泄露是指: 神经程序修复系统在基准测试集上的某条可修复的缺陷-补丁对的缺陷部分出现在了训练数据当中, 即该条可修复的缺陷-补丁对中的缺陷部分出现在了某一条训练数据的缺陷代码中.
- 补丁代码泄露(fixed code). 该类型的数据泄露是指: 神经程序修复系统在基准测试集上的某条可修复的缺陷-补丁对的补丁部分出现在了训练数据当中, 即该条可修复的缺陷-补丁对中由神经程序修复系统生成的补丁出现在了某一条训练数据的补丁代码中.

此处的“出现”是指: 基准测试集上的缺陷-补丁对(或缺陷-补丁对中的缺陷、补丁部分)与训练数据中的缺陷-补丁对字符串完全匹配(除去空格、空行以及注释等不含代码信息的字符), 或为子串关系, 即基准测试集上的数据是某条训练数据的子串. 选择字符串完全匹配(type-1 匹配)作为判定数据泄露的原因是, 我们认为: 只有当字符串完全匹配时, 才会使得神经程序修复系统在进行测试之前就已经完全学习到某一测试样本的语义、句法、代码结构以及从缺陷到补丁的转换范式等修复信息. 这使得神经程序修复系统在测试时遇到相同样本的情况下, 仅通过对该样本修复信息的记忆来输出补丁. 我们认为: 这种学习方法没有使得神经程序修复系统理解该样本的修复范式, 而是对修复信息的简单记忆. 而其余类型的代码匹配, 我们则认为不会引入狭义的数据泄露. 表 1 是几种代码匹配类型对应的含义[27].

表 1 代码匹配类型及其定义

名称	类型	定义
Type-1 匹配	句法匹配	除去空格,空行和注释后,两段代码相同
Type-2 匹配	句法匹配	除去对标识符(函数名、类名和变量名等)的修改, 两段代码相同
Type-3 匹配	句法匹配	片段被部分重排序、增加或删除, 修改后, 两段代码相同
Type-4 匹配	语义匹配	语义相同但句法上没有联系

以上是数据泄露的分类与定义, 对于不同的神经程序修复系统, 对应的数据泄露类型可能不同; 而对于不同的数据泄露类型, 其检测的方式也不相同. 我们将在第 3.5 节对此展开详细的介绍.

1.3 神经程序修复系统的鲁棒性及其评估方法和评价指标

1.3.1 神经程序修复系统鲁棒性的定义

深度学习模型的鲁棒性通常被定义为对输入产生微小变化, 输出保持不变的能力. 现有的深度学习模型普遍存在鲁棒性较差的问题. 例如: 对于卷积神经网络(convolutional neural network, CNN), 在对输入的图片加入对抗扰动后, 卷积神经网络便不能再次识别该图片[28]; 对于神经机器翻译模型(neural machine translation, NMT)[29], 在对输入语句做出极小改变后, 可能会引起翻译结果的剧烈改变[30]. 对于神经程序修复系统, 其鲁棒性则被引申为: 对于语义一致的输入, 神经程序修复系统生成语义一致的输出的能力. 而语义一致性的定义是: 对于两段语义一致的程序, 针对域中的任何输入, 程序的行为或输出结果是一致的[31]. 神经程序修复系统鲁棒性和语义一致性的定义可被表示为以下公式.

$$\forall p,p_t\in P:\ p=p_t,\ NPR(p)=NPR(p_t) \tag{1}$$