doi: 10.12194/j.ntu.20240117001
引文格式: 石翔宇, 鞠小林, 陈翔. 基于缺陷报告降噪和抽象语法树表示的软件缺陷定位方法[J]. 南通大学学报(自然科学版), 2024, 23(3):23-33.

基于缺陷报告降噪和抽象语法树表示的软件缺陷定位方法

石翔宇, 鞠小林*, 陈 翔
(南通大学 人工智能与计算机学院, 江苏 南通 226019)

摘要:自动化的缺陷定位方法能够加快程序员利用缺陷报告定位到复杂软件系统缺陷代码的过程。现有相关缺陷定位方法存在两方面问题:忽略了缺陷报告中噪音信息的影响;程序代码表示的过程中丢失了大量的上下文结构信息。为了解决上述问题,提出一种自动化缺陷定位方法 BRFN(bug report fault localization)。该方法首先利用双向的信息传播机制对程序的抽象语法树进行编码;接着针对缺陷报告使用 TextCNN 和注意力机制学习与缺陷相关的特征;最后通过计算缺陷报告和源代码文件之间的相关性,开展缺陷定位,并基于 4 个广泛用于缺陷定位研究的软件项目评估 BRFN 方法的有效性。实验结果表明,相较于 BugLocator、LS-CNN 和 CAST 现有的缺陷定位方法,BRFN 在多个评价指标上均取得了更好的效果。具体而言,BRFN 在 4 个开源项目上的 Acc@1、MRR 和 MAP 性能平均提升了 56.3%、43.4%和 46%。此外,进一步设计消融实验来验证 BRFN 中各模块的贡献。结果表明,缺陷报告降噪策略和双向信息传播策略可以提升缺陷定位的准确性。

关键词:缺陷定位;深度学习;信息检索;注意力机制;程序表示学习
中图分类号: TP311.52 **文献标志码**: A **文章编号**: 1673-2340(2024)03-0023-11

Software defect localization method based on defect report denoising and abstract syntax tree representation

SHI Xiangyu, JU Xiaolin*, CHEN Xiang
(School of Artifical Intelligence and Computer Science, Nantong University, Nantong 226019, China)

Abstract: Automated defect localization methods can accelerate the process by which programmers use defect reports to pinpoint defect code in complex software systems. Existing defect localization methods face two main issues: neglecting the impact of noisy information in defect reports and losing significant contextual structural information during code representation. To address these issues, a novel automated defect localization method, named BRFN (bug report fault localization), is proposed. This method first encodes the abstract syntax tree of the program using a bidirectional information propagation mechanism. It then employs TextCNN and attention mechanisms to learn defect-relevant features from defect reports. Finally, it calculates the correlation between defect reports and source code files to perform defect localization. The effectiveness of the BRFN method is evaluated based on four widely used software projects for defect localization research. Experimental results show that BRFN outperforms existing methods such as BugLocator, LS-CNN, and CAST across multiple evaluation metrics. Specifically, BRFN improves Acc@1, MRR, and

收稿日期: 2024-01-17 **接受日期**: 2024-02-21
基金项目: 国家自然科学基金面上项目(61673384);江苏省现代教育技术研究项目(2022-R-98984)
第一作者简介: 石翔宇(1997—), 男, 硕士研究生。
* **通信联系人**: 鞠小林(1976—), 男, 副教授, 博士, 主要研究方向为软件质量保证、智能软件工程。E-mail:ju.xl@ntu.edu.cn

MAP by 56.3%, 43.4%, and 46%, respectively, on four open-source projects. Additionally, ablation experiments are conducted to validate the contribution of each module in BRFN. The results indicate that both the defect report denoising strategy and bidirectional information propagation strategy enhance the accuracy of defect localization.

Key words: defect localization; deep learning; information retrieval; attention mechanism; program representation learning

缺陷定位旨在通过分析程序代码及行为,识别导致软件失效的缺陷所在位置。在缺陷定位过程中,开发人员通常需要结合多种技术和工具,以便更准确地识别和定位缺陷。其中,程序分析和缺陷报告分析是两种常用的技术[1]。程序分析是一种通过对源代码进行静态或动态分析,以获取有关程序行为和性能信息的技术。它可以帮助开发人员更好地理解程序,并找到导致缺陷的根本原因。缺陷报告是软件测试人员用于报告程序中存在问题的文档,其中包含了运行环境信息、测试用例、问题的描述、复现步骤等。结合程序分析和缺陷报告分析可以提高缺陷定位的准确性和效率。具体而言,开发人员可以使用程序分析技术来分析源代码文件,以识别软件中潜在的缺陷点,并使用缺陷报告来验证这些缺陷点是否与实际问题相关。通过这种方式,软件调试人员可以更快速地定位和修复缺陷。

基于信息检索的缺陷定位方法将缺陷报告视作查询,源代码文件视作文档,通过计算查询与各个文档之间的相似度,确定哪些源代码文件最有可能与当前缺陷报告相关[2-3]。随着自然语言处理领域的不断发展,研究人员尝试将各种深度神经网络和信息检索方法结合起来定位软件缺陷[4-6]。这类方法使用缺陷报告及其相关源代码文件组成的数据集训练神经网络模型以捕获一些高层次抽象特征,从而取得不错的缺陷定位效果。

影响基于信息检索的缺陷定位方法性能的其中一个重要因素是查询质量的好坏。在实际软件开发过程中,撰写缺陷报告的对象可以是软件测试人员或者是普通用户, 他们编写缺陷报告的水平不同,缺陷报告的质量可能会存在较大的差异。对于低质量的缺陷报告,除了少部分内容对定位缺陷文件有帮助外,剩余部分都可以被归为数据噪声。而这些数据噪声可能会影响缺陷定位的效果。

图 1 是 AspectJ 项目一个简化的缺陷报告[7],该缺陷报告的描述部分指出:在将 2 个 aspect 编译到一个 jar 包时出现了问题,并希望相关人员给出一些修复建议。虽然详细的缺陷报告可以帮助维护人员在手动修复缺陷时更快地定位缺陷文件的位置,但是对于自动化的缺陷定位工具,除了红色标注的部分可以成为定位的关键信息,剩余的部分可能是影响定位性能的噪音。现有的基于信息检索的缺陷定位方法通常将缺陷报告的摘要和描述部分直接作为模型的输入,缺少对输入的降噪处理。如果查询中包含过多的噪音信息,会在缺陷报告和源代码文件之间建立错误关联,影响定位准确性。

程序语言表示的源代码文件是结构化的,如何从程序语言中提取与缺陷报告关联的结构信息,再将这些信息用嵌入向量表示是当前缺陷定位方法所面临的挑战之一。

Bug report for AspectJ project

Bug ID: 125101

Summary: StackOverFlow in ReferenceType.isAssignableFrom()

Description: Hi. I am weaving two aspects into a jar file "jar-to-jar". ajc 1.5.0 gives me a huge error message (see attachement). Would be great if I could get any recommendations on this. Thanks! Eric

Report Time: 2006/1/24 19:40:36
Status: resolved fixed

图 1 125101 号缺陷报告

Fig. 1 Bug report #125101

许多缺陷定位方法表明,程序语言可以从抽象语法树(abstract syntax tree,AST)的语法和结构化表示中受益。Liang 等[6]在定制的 AST 上应用 TBCNN 模型[8],利用一组固定深度的特征提取器在整个 AST 上滑动提取局部信息,再采用自底向上的编码层对局部信息进行整合,进而获得源代码的向量表

示。Tai 等[9]针对 AST 树状结构提出了 Tree-LSTM (tree long short term memory)模型，该模型通过自底向上的信息传播方式，将AST 中每个节点的当前输入与其子状态结合，更新所有节点的向量表示，最终采用根节点作为源程序的向量表示。

尽管上述基于 AST 的程序表示方法在缺陷定位任务上取得了一定的性能提升，但仍存在改进空间。例如这些网络中的节点信息更新方式均为单向的自底向上传递，低层叶子节点难以获得足够的上下文信息，因此根据这些节点获得源代码的最终向量表示时会导致信息丢失。

上述提到的缺陷定位方法存在的不足可以归纳为以下 2 个问题：a)如何对缺陷报告进行降噪以提取与缺陷相关的信息；b)如何更好地表示程序代码的结构信息，在代码向量表示的过程中保留源代码的上下文信息。

本文提出 BRFN(bug report fault localization)以解决上述 2 个问题。针对问题 a)，利用注意力机制，重点关注缺陷报告中和定位相关的信息以优化缺陷定位。具体而言，BRFN 首先对缺陷报告和源代码文件进行文本预处理，再通过嵌入层编码得到两者的向量表示，接着使用 TextCNN 网络分别提取文本层级的特征，最后使用注意力机制对缺陷报告的特征重新分配权重，让模型聚焦于缺陷报告中的关键信息，给噪音部分分配较少的关注度。针对问题 b)，提出基于递归树状结构神经网络 Tree-transformer[10]的源代码向量表示方法。具体而言，BRFN 首先将源代码转换成其对应的 AST 结构，然后对 AST 中的节点进行嵌入编码，接着通过自底向上和自顶向下 2 轮信息传播来更新所有节点的向量表示，最后在整棵 AST 上使用全局池化获得源代码的向量表示。

本文的主要贡献如下：

1)从分析缺陷报告的角度出发，提出一种联合降噪策略。该策略通过 TextCNN 特征提取，结合多头注意力机制，可以在保留有效缺陷定位线索的同时降低噪音对缺陷定位性能的影响。

2)从程序代码表示入手，利用双向信息传播机制挖掘程序 AST 中的深层次结构信息，丰富代码表示上下文信息，从而提升缺陷定位的准确性。

1 相关工作

1.1 基于信息检索的软件缺陷定位

缺陷定位是将自然语言编写的缺陷报告和程序语言编写的源代码文件进行匹配的过程，类似于一个信息检索任务。源代码文件的可疑程度按照其与缺陷报告的相关性排序。相关性可以通过直接和间接的方式进行度量。直接方式通过计算缺陷报告和源文件之间的文本相似性度量，如 Zhou 等[2]提出 BugLocator 方法，在矢量空间模型中计算文本相似度对文件进行排名；间接方式通常包括根据项目元数据获取缺陷修复频率、类名相似性和其他特征[5]。一些研究人员通过从语义角度提取文本的语义特征来计算相似性。这种方法主要借助神经网络模型自动提取深度语义特征[4-6]。

只有当源代码文件中包含缺陷报告提到的类名或标识符名这些用于定位的提示信息时，文本相似性特征才会有帮助，对于那些缺乏提示信息的缺陷报告，浅层次的术语匹配无法实现准确定位。本文不仅考虑了缺陷报告和源代码文件之间的术语匹配，还从源代码的抽象语法树结构中挖掘深层次结构信息，通过融合这两方面的信息，提高缺陷定位的效率。

1.2 注意力机制

注意力机制最早被引入神经机器翻译领域[11]，随后成为神经网络结构中越来越常见的组成部分，并应用于各种任务，例如图像标题生成[12-14]、文本分类[15-16]、动作识别[17]。注意力机制可以作为一种资源分配方案，通过在每个时间步给予输入中的不同部分不同程度的关注，使模型能够更聚焦在任务相关的重要信息上[18]。Xiao 等[19]提出 ALBFL 利用 Transformer[20]编码器从源代码中学习语义特征，并通过自注意力机制整合静态统计特征和动态特征。这种注意力机制有助于模型聚焦关键特征，可以有效地对缺陷代码文件进行排序。

对于较长的文本序列输入，注意力机制的关注点不仅限定在某个固定长度的局部窗口内，而且可以扩大到更为全局的范围，这使得模型可以获知整个输入序列的上下文信息，而不仅仅是局部的信息。这种扩大的“视野”在处理长序列中的长期依赖

关系时尤其有效。例如 Wang 等[21]在文本分类任务中,将注意力机制与 LSTM[22]模型相结合,发现与单独使用 LSTM 相比,结合注意力机制的 LSTM 网络明显提高了分类的性能。

在利用注意力机制进行降噪方面,Zhang 等[23]提出了 AttL 编码器,这是一种带有注意力机制双向长短期记忆网络,可以捕捉缺陷报告中的语义信息并自动聚焦于最相关的内容。AttL 编码器通过软注意力机制全面考虑缺陷报告中的单词,为对任务至关重要的单词分配更高权重;对与任务无关的单词则分配较低权重。AttL 编码器能够降低缺陷报告的噪音,提取缺陷报告的关键语义信息。此外,Han 等[24]提出 bjXnet 模型,以缺陷报告、源代码文本和 CPG 为输入,映射到相同的潜在语义空间。通过多模态表示学习和注意力机制,对非缺陷图特征降噪,集中关注和定位任务相关的源代码特征。最终,通过比较余弦相似度表征报告与源代码的关联性。

本文利用注意力机制对缺陷报告的特征重新分配权重,让模型聚焦于缺陷报告中的关键信息实现降噪。在双向信息传播的过程中(见 2.2.2 和 2.2.3 部分)注意力机制被用来传递子节点之间的信息及父节点和子节点之间的信息。

1.3 代码向量表示

代码表示的目的是自动地从程序中学习有用的特征,将特征表示为低维稠密向量,高效地提取程序语义并应用于下游任务[25-26]。程序表示方法可分为 3 类[27]:基于数据的表示方法[28]、基于序列的表示方法[29]和基于图的表示方法。

基于数据的表示方法假定程序是将输入映射到输出的函数,使用程序在运行时的输入和输出数据来学习其向量表示[28]。基于序列的表示方法将程序视为自然语言,将 NLP 中使用的模型直接应用到源代码中。例如 Xiao 等[29]使用 Word2Vec[30]词嵌入模型来表示程序代码,以捕捉缺陷报告和源代码文件中的语义信息,然后利用增强的卷积神经网络进行缺陷定位。Wang 等[26]使用预训练的 BERT 模型[31]对程序代码进行预训练和微调从而获得向量表示。

基于图的表示方法主要利用了抽象语法树(abstract syntax tree,AST)、程序依赖图(program dependency graph,PDG)、数据流图(data flow graph,DFG)和控制流图(control flow graph,CFG)等结构。相较基于序列的表示方法,基于图的表示方法能包含更多的代码结构信息和语义信息。例如 Lou 等[32]采用基于图的表示形式将详细的测试覆盖范围信息和代码结构整合为一幅图,并利用门控图神经网络从基于图的覆盖率表示和程序实体排序中学习有价值的特征。Tian 等[33]使用 AST 和 Tree-LSTM 从代码片段中挖掘深层语义,实现了准确的函数级缺陷定位。基于 AST 的代码向量表示方法[32-33]有助于从代码中提取上下文信息及引入分层依赖关系,可以改善代码理解并捕获深层语义特征,从而更准确地辅助定位缺陷。

本文在代码向量表示的过程中,首先,将源代码转换为 AST 结构;然后,对 AST 节点进行嵌入编码;接着,通过自底向上和自顶向下两轮信息传播更新节点向量表示;最后,采用全局池化在整棵AST上获取源代码的向量表示。

2 方法设计

本节介绍BRFN 方法的设计和实现,整体框架见图 2。针对缺陷报告,首先利用 TextCNN[34]提取语义特征,然后通过注意力机制学习关键信息;针对源代码文件,首先解析为抽象语法树,在语法树上双向信息传递进行程序表示学习。最后通过学习两者向量间的隐含关系,推荐可能存在缺陷的源代码文件。

2.1 缺陷报告降噪

为了从缺陷报告中提取对缺陷定位有用的信息,同时降低噪音信息对缺陷定位产生的负面影响,BRFN 针对缺陷报告设计了降噪网络,如图 2 的缺陷报告降噪部分所示。该网络首先使用 TextCNN 对缺陷报告和源代码文件提取局部关键信息, 然后通过注意力机制学习如何根据源代码的关键定位信息,对缺陷报告的向量表示重新分配重要性权重。

2.1.1 *编码层*

首先对缺陷报告和源代码文件进行文本预处理,包括根据“驼峰拼写”规则拆分复合词,过滤源代码文件中的“public”“void”等关键词。这些关键词无法提供有效的定位信息,而且会占用大量计算资源。文本预处理后,缺陷报告和源代码文件都被

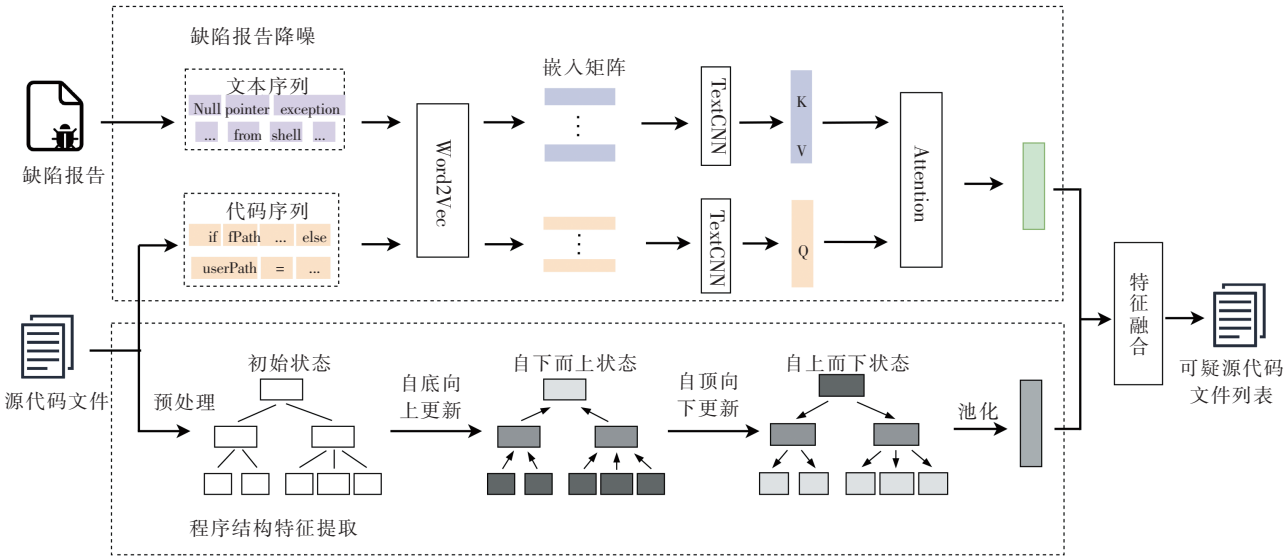

图 2　BRFN 整体框架

Fig. 2　Overall framework of BRFN

表示为单词序列。然后将单词序列编码为嵌入向量。与使用独热编码(One-hot encoding)表示单词向量相比，Word2Vec[30]将每个词表示为稠密的实数向量，从而大幅降低嵌入向量的维度。此外，Word2Vec在特定语料库上预训练得到，其词向量表示蕴含丰富的语义信息，含义相近的词汇在向量空间中距离也更为接近。选择 Word2Vec 获得单词的预训练向量，将每个单词表示为一个 300 维的向量。

对于每一个缺陷报告和源代码文件序列，固定序列的最大单词数，对数量不足的部分随机初始化一个 300 维的向量补齐，对超过的部分进行截断，最后包含 b 个单词的缺陷报告和 s 个单词的源代码文件被分别表示为 $b \times 300$ 和 $s \times 300$ 大小的嵌入矩阵，接下来将嵌入矩阵送入卷积层做特征提取。

2.1.2　特征提取

TextCNN[34]是一种针对文本任务设计的卷积神经网络，具有较强的局部信息提取能力。TextCNN 通过在文本向量上滑动卷积核，可以实现类似 N-gram[35]的功能，从而捕获文本之间的上下文信息。

对于自然语言表示的缺陷报告，能够传递语义信息的基本单元是一个单词或术语。对于编程语言表示的源代码文件，程序语句是传递语义信息的基本单元，编程语言的语义必须从多个语句的语义及这些语句在其执行路径上的相互作用中推断出来[4]。为了从缺陷报告和源代码文件中提取和缺陷有关的信息，基于 TextCNN 模型设计卷积神经网络分别对缺陷报告和源代码文件进行特征提取，为缺陷报告使用单层卷积，对源代码文件使用双层卷积。第一层用于捕获词汇间的特征，第二层用于捕获句子之间的语义。

式(1)和式(2)是 TextCNN 对输入序列进行一次卷积操作的过程。首先 K 个不同窗口大小的卷积核 Conv_i(其中 $i \in [1,k]$)被用来在单词向量 *word_vector* 上滑动以生成 K 个维度的特征映射，随后在每个特征映射上应用最大池化操作 MaxPool，获得当前卷积核 Conv_i 下的特征向量 $\boldsymbol{X}_i$，最后将所有特征向量拼接，得到文本特征 *Text_feature*。

$$\boldsymbol{X}_i = \mathrm{MaxPool}\{Conv_i(word\_vector)\}, i = 1,2,\cdots,k, \tag{1}$$

$$Text\_feature = \mathrm{Concat}(\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_k)。 \tag{2}$$

对于源代码文件，将第一层卷积输出的文本特征 *Text_feature* 作为第二层卷积的输入，再按照式(1)和式(2)进行第二次卷积，得到源代码文件的语句特征 *Smt_feature*。

2.1.3　联合降噪

通过 TextCNN 特征提取器，缺陷报告和源代码文件在不同的层级上被表示为特征向量。然而缺陷报告的特征向量中并非所有的部分都与缺陷定位任务相关。为了使模型专注于与缺陷定位任务相关

的信息,采用多头注意力机制对缺陷报告和源代码文件进行联合降噪。具体而言,将 2.1.2 中缺陷报告的文本特征向量 *Text_ feature* 作为注意力机制中的键(*Key*,K)和值(*Value*,V),源代码文件的语句特征 *Sent_ feature* 作为查询(*Query*,Q)。注意力机制将集中在源代码文件上,以便更好地理解缺陷报告中的关键信息。

计算 Q 和 K 之间的点积,得到每个源代码文件 Q 与缺陷报告 K 上不同部分的相似度分数,通过 Softmax 函数将相似度分数转化为注意力权重,同时确保注意力权重的总和为 1, 表示每个位置的相对重要性。表达式如下:

$$A = \mathrm{Softmax}(\frac{Q \times K^{\mathrm{T}}}{\sqrt{d_K}})。\tag{3}$$

利用式(3)得到的注意力权重对缺陷报告的原始特征向量 V 进行加权汇聚,如式(4)所示,得到注意力汇聚后的缺陷报告表示 φ_{br},其中 V_i 代表缺陷报告的每一部分值向量。

$$\varphi_{\mathrm{br}} = \sum_i A_i(Q,K)V_i。\tag{4}$$

φ_{br} 向量根据对定位缺陷所提供的重要性重新分配了关注度,对缺陷报告中的噪音信息进行了过滤,包含缺陷报告和源代码之间的关系。

2.2 程序结构特征提取

为了能够从程序代码中提取有用的特征,同时将特征表示为低维稠密向量。在树状结构神经网络 Tree-transformer[10]的基础上学习源代码的向量表示,如框架图 2 的程序结构特征提取部分所示。首先源代码文件被预处理为对应的 AST 结构,然后对 AST 中的节点进行嵌入编码,随后通过自底向上和自顶向下两轮信息传播来更新所有节点的状态,最后在整棵 AST 上使用全局池化得到源代码的向量表示。

2.2.1 源代码预处理

在源代码预处理环节,源代码文件被编译成抽象语法树 AST,每一个 AST 是由叶子节点 N_{leaf} 和非叶子节点 $N_{\mathrm{non\_leaf}}$ 组成的点集 $N = \{N_{\mathrm{leaf}}, N_{\mathrm{non\_leaf}}\}$。对点集 N 采取 2.1.1 中同样的预处理操作,用Word2Vec 将每一个节点编码为初始向量$\{e_i \mid i \in N\}$,该初始向量会在接下来的自底向上和自顶向下传播过程中被不断更新。

2.2.2 自底向上更新

自底向上更新的目的是将上下文信息从叶子节点聚合到根节点。通过自底向上更新,每一个节点的初始向量$\{e_i \mid i \in N\}$均被更新为自底向上状态$\{h_{i\uparrow} \mid i \in N\}$。以非叶子节点 n_i 为例,如果 n_i 的子节点 $c_i = \{c_1, c_2, \cdots, c_n\}$(其中下标 n 为子节点的个数)均为叶子节点,就采用初始状态 e_{c_i} 作为所有子节点 c_i 的自底向上状态 $h_{c_i\uparrow}$,然后通过式(5)更新节点 n_i 的自底向上状态 $h_{i\uparrow}$,其中 θ 为权重系数。

$$h_{i\uparrow} = f(e_i, h_{c_i\uparrow}, \theta),\tag{5}$$

式中的 f 函数由两层注意力机制构成。其中第一层自注意力机制在子节点 $c_i = \{c_1, c_2, \cdots, c_n\}$ 上建模子节点间的依赖关系。自注意力机制中的 Q, K, V 均来自子节点的自底向上状态 $h_{c_i\uparrow}$。第二层多头注意力机制用于捕获父节点 n_i 及其子节点 c_i 之间的依赖关系。其中,父节点 n_i 的初始向量 e_i 被当作Q,子节点在第一层自注意力机制得到的输出当作 K 和 V。经过自底向上传播后,所有节点的初始向量$\{e_i \mid i \in N\}$均被更新为自底向上状态$\{h_{i\uparrow} \mid i \in N\}$。

2.2.3 自顶向下更新

在节点的自底向上状态$\{h_{i\uparrow} \mid i \in N\}$上再执行自顶向下的信息传播。在自顶向下信息传播过程中,所有节点会将信息从父节点传递给子节点,同时所有节点被更新为自顶向下状态$\{h_{i\downarrow} \mid i \in N\}$。

以根节点 n_i 更新其子节点 $c_i = \{c_1, c_2, \cdots, c_n\}$ 的自底向上状态 $h_{i\uparrow}$为例。由于 n_i 是根节点,就用根节点的自底向上状态 $h_{i\uparrow}$ 作为其自顶向下状态 $h_{i\downarrow}$,即 $h_{\mathrm{root}\downarrow} = h_{\mathrm{root}\uparrow}$。接着使用一层多头注意力机制将信息从父节点 n_i 传递给子节点 $c_i = \{c_1, c_2, \cdots, c_n\}$。与2.2.2 节中的多头注意力机制不同的是,在这里子节点 c_i 的自底向上状态 $h_{c_i\uparrow}$ 被作为查询 Q,父节点的自顶向下状态 $h_{i\downarrow}$ 被当作 K, V。由于选取的 K, V 的数目为 1,在计算注意力分数时会导致 Softmax 函数失效,因此直接将父节点 n_i 的自顶向下状态 $h_{i\downarrow}$ 加到子节点$\{c_1, c_2, \cdots, c_n\}$的自底向上状态上得到注意力分数。

2.2.4 整棵 AST 的向量表示

节点信息双向传播完成后，每个节点都被更新为自顶向下状态$\{h_{i\downarrow} \mid i \in N\}$，该状态包含了从其他节点获取的上下文信息，从而能够捕获全局依赖关系。在所有节点的自上而下状态上使用全局池化函数可获得整棵语法树的最终表示。具体表示为

$$\varphi_{\mathrm{sf}} = \sum_{i \in N} \mathrm{Softmax}(W_{\mathrm{pooling}} h_{i\downarrow}) \odot h_{i\downarrow}, \tag{6}$$

式中：W_{pooling}是池化过程中的权重系数；⊙表示元素对应位置相乘。最终整棵抽象语法树的向量表示作为该源代码文件的嵌入向量表示φ_{sf}。

2.3 特征融合

在对缺陷报告r_i的信息降噪，进而提取源代码文件s_j的特征，得到一组缺陷报告和源代码文件的嵌入向量表示$(\varphi_{\mathrm{br}}^{i}, \varphi_{\mathrm{sf}}^{j})$之后，需要进行特征融合。将特征向量表示$(\varphi_{\mathrm{br}}^{i}, \varphi_{\mathrm{sf}}^{j})$送入由全连接网络构成的特征融合层，计算缺陷报告和源代码文件之间的相关性。单个(r_i, s_j)对的相关性预测分数通过式(7)对特征向量加权得到。

$$F(r_i, s_j) = \sigma(w_1 \varphi_{\mathrm{br}}^{i} + w_2 \varphi_{\mathrm{sf}}^{j} + b), \tag{7}$$

其中：σ表示激活函数，这里使用 Softmax 函数；w_1和w_2表示权重矩阵；b表示函数偏置。使用交叉熵损失函数l计算预测的相关性结果$F(r_i, s_j)$与真实标签y_{ij}之间的损失，表示为

$$l = -(y_{ij} \ln(F(r_i, s_j)) + (1 - y_{ij}) \ln(1 - F(r_i, s_j))). \tag{8}$$

最终通过式(9)基于随机梯度下降(stochastic gradient descent，SGD)的目标函数更新模型中的所有参数，其中λ为调节系数，$\| w \|$是正则化项。

$$L = \sum_{i,j} l(r_i, s_j, y_{ij}) + \lambda \| w \|^2。 \tag{9}$$

3 实验与结果

3.1 实验设计

3.1.1 研究问题

本文设计了以下 2 个研究问题(RQ)来评估 BRFN 的有效性及 BRFN 中各模块的贡献。

RQ1：与其他基线方法相比，BRFN 的有效性如何？为评估 BRFN 的缺陷定位效果，将该模型与 BugLocator、LS-CNN 和 CAST 方法进行比较。

BugLocator[2]是一种基于信息检索技术的缺陷定位方法，它设计了一个向量空间模型，该模型通过评估源代码文件和缺陷报告之间的相似性，来识别和缺陷报告相关的源代码文件。

LS-CNN[36]使用 CNN 提取缺陷报告的特征，同时结合 CNN 和 LSTM 从源代码中捕获序列语义和结构信息。

CAST[6]是一种结合树形卷积神经网络与定制的抽象语法树的方法，它捕获缺陷报告和源文件中的词汇语义及源代码中的层次结构信息。

RQ2：本文提出的缺陷报告降噪策略及双向信息更新策略分别能带来多少效果上的提升？

通过消融实验探讨降噪策略、自底向上传播和自顶向下传播对模型的贡献。

3.1.2 数据集

本文选择 Ye 等[3]提供的数据集，该数据集利用 Bugzilla 缺陷管理系统和 Git 版本控制系统从开源项目中构建。由于该数据集中 Birt 和 JDT 项目的文件数量过多，为节省实验时间，只选取了其中规模相对较小的 4 个开源项目：Eclipse、SWT、Tomcat 和 AspectJ，包含对应的缺陷报告、源代码文件、API 文档及缺陷报告与相关源代码文件之间的映射。数据集详细信息描述如表 1 所示。根据缺陷报告的提交时间，将缺陷报告按升序排序，划定前 80%的数据用于模型训练，剩余 20%的数据均分用于模型测试和模型验证。

表 1 数据集的详细信息

Tab. 1 Details of the dataset

项目	缺陷报告数	源代码文件数	平均缺陷文件数
Eclipse	6 495	3 454	2.5
SWT	4 151	2 056	2.0
Tomcat	1 056	1 552	2.6
AspectJ	593	4 439	3.7

3.1.3 评价指标

采用 Acc@k、MRR 和 MAP 3 种指标来评估 BRFN 的性能，这些指标被广泛用于评价缺陷定位效率。

Acc@k用于衡量模型推荐的准确性。给定一个缺陷报告，如果推荐列表的前k个结果中至少出现

一个与当前缺陷报告相关的缺陷源代码文件,则当前缺陷报告被视作成功定位。在实际缺陷定位场景中,95%以上的软件维护人员通常仅检查推荐列表中的前 10 个文件[2]。实验时,k 的值通常取 1、5 和10。

MRR(mean reciprocal rank)是一种统计度量。在单个查询中,RR(reciprocal rank)是首个被正确定位的源代码文件在推荐列表中排名的倒数,MRR 是所有查询倒数排名 RR 的平均值,表达式为

$$MRR = \frac{1}{|B|}\sum_{i=1}^{|B|}\frac{1}{rank_i}, \tag{10}$$

其中:B 是缺陷报告的集合;$|B|$为缺陷报告的数目;$rank_i$表示根据第 i 个缺陷报告成功推荐出的首个错误源代码文件在推荐列表中的位置。

MAP(mean average precision)考虑可能会出现一个缺陷报告与多个源代码文件相关联的情况,首先计算平均精度 AvgP(average precision),AvgP 度量单个缺陷报告的平均定位准确率,定义为

$$AvgP = \frac{1}{|C|}\sum_{k=1}^{|C|}\frac{k}{rank_k}, \tag{11}$$

其中:C 表示缺陷源代码文件的集合;$|C|$是缺陷源代码文件的数量;$rank_k$表示第 k 个缺陷源代码文件在推荐列表中的排名。然后计算 AvgP 的平均值得到MAP 为

$$MAP = \sum_{i=1}^{|B|}\frac{AvgP_i}{|B|}, \tag{12}$$

其中:B 为缺陷报告的集合;$|B|$表示缺陷报告的数目;$AvgP_i$表示第 i 个缺陷报告的定位准确率。

3.1.4 实验运行环境

实验环境设置为:CPU AMD 5900X,GPU NVIDIA GeForce RTX3090 24 GB 内存。使用 Python3.6 作为编程语言,基于 Pytorch1.13 框架搭建神经网络。在训练阶段,epoch 设置为 100,Batch size 设置为 32,优化器选取 Adam,Learning rate 设置为 1×10^{-3}。

3.2 实验结果与分析

3.2.1 RQ1 的实验结果与分析

该研究问题旨在对比 BRFN 与其他基线方法的有效性。在 4 个开源数据集上评估了 BRFN,并与其他 3 种有代表性的软件缺陷定位技术进行了比较。

表 2 展示了 BRFN 与基线方法在 4 个数据集上的性能比较结果。BRFN 在所有项目上的平均 Acc@10、MRR、MAP 分别达到了 0.90、0.56 和 0.47,其中 Acc@10 最高达到了 0.92。表明了 BRFN 在缺陷定位上的有效性。

表 2 BRFN 与 3 种基线方法的比较

Tab. 2 Comparison between BRFN and three baselines

项目	方法	Acc@1	Acc@5	Acc@10	MRR	MAP
Eclipse	BugLocator	0.29	0.46	0.81	0.47	0.34
	LS-CNN	0.37	0.44	0.86	0.51	0.32
	CAST	0.33	0.49	0.89	0.49	0.41
	BRFN	0.45	0.58	0.90	0.58	0.49
SWT	BugLocator	0.27	0.45	0.85	0.32	0.28
	LS-CNN	0.31	0.50	0.84	0.43	0.37
	CAST	0.35	0.49	0.87	0.48	0.39
	BRFN	0.43	0.55	0.91	0.51	0.42
Tomcat	BugLocator	0.23	0.37	0.82	0.40	0.43
	LS-CNN	0.34	0.54	0.82	0.41	0.34
	CAST	0.37	0.59	0.83	0.46	0.36
	BRFN	0.45	0.53	0.89	0.59	0.45
AspectJ	BugLocator	0.25	0.42	0.83	0.36	0.33
	LS-CNN	0.32	0.37	0.80	0.42	0.37
	CAST	0.30	0.33	0.83	0.49	0.44
	BRFN	0.37	0.47	0.92	0.57	0.53

与 BugLocator、LS-CNN、CAST 3 种模型对比,BRFN 在各项性能指标上均表现出显著的提升。相对于 BugLocator,BRFN 的 Acc@1 提高了 48%~95%,MRR 提高了 23%~59%,MAP 提高了 4%~60%;与 LS-CNN 相比,BRFN 的 Acc@1 提高了 15%~38%,MRR 提高了 13%~43%,MAP 提高了 13%~53%;与 CAST 相比,BRFN 的 Acc@1 提高了 21%~36%,MRR 提高了 6%~28%,MAP 提高了 7%~25%。

BRFN 性能的提高主要源于两方面的创新。首先,BRFN 在提取缺陷报告特征时采用联合源代码的文本序列信息进行降噪,从而有效减少了噪音在模型中的传播;其次,BRFN 从程序 AST 中挖掘结构特征,相较于 LS-CNN 仅从文本层面处理源代码,BRFN 可以获得更为丰富的结构信息。这两个方面的创新使得 BRFN 能够更全面、准确地捕捉源代码中的关键信息,提高了缺陷定位的准确性。尽管 CAST 同样从 AST 角度表示源代码,但其采用的单向自底向上的信息传播方式未能有效集成上下文信息。

对 RQ1 的研究结论如下:相较于 BugLocator、

LS－CNN、CAST 3 种缺陷定位方法，BRFN 在 Acc@1、MRR 和 MAP 上均有显著提升。平均而言，BRFN 在4 个开源项目上的 Acc@1、MRR 和 MAP 分别实现了56.3%、43.4%和 46%的性能提升。

3.2.2 RQ2 的实验结果与分析

该研究问题旨在探索本文提出的缺陷报告降噪策略及双向信息更新策略分别能带来多少效果上的提升。

首先，通过消融实验进一步探究降噪处理策略对缺陷定位产生的影响，包括：1）不使用降噪策略；2）使用降噪策略，但不使用自顶向下更新策略。不使用降噪策略指的是通过 TextCNN 提取的缺陷报告特征向量不做降噪处理直接输入特征融合层，用于预测相关性分数。不使用自顶向下更新策略是指 AST 的节点信息在自底向上更新完成之后，直接通过池化操作获得整棵语法树的特征向量。其次，通过消融实验研究双向信息传递机制的有效性。

实验结果如表 3 所示。表中 BRFN w/o Denoise 表示不使用降噪策略处理缺陷报告，BRFN w/o Top-down 表示不使用自顶向下更新策略。从表 3 可以看出，如果在缺陷报告处理中不使用降噪策略，所有项目的 Acc@10、MRR 和 MAP 都出现了不同程度的下降。在 Eclipse 项目上，与使用降噪策略的方法相比，不使用降噪策略在 Acc@10 准确率上出现了最大幅度 7.8%的性能下降。研究表明，引入降噪策略来处理质量参差不齐的缺陷报告确实是必要的。但是在 Tomcat 和 AspectJ 项目上，Acc@10 准确率的下降程度并不明显，可能的原因在于这 2 个项目规模较小，提交的缺陷报告主要来自专业维护人员，同时说明即使在相对质量较高的报告集中，降噪策略仍对性能产生一定影响。根据实验结果还可以观察到，在去除自顶向下的更新策略后，BRFN 在所有项目上的各项指标显著下降，Acc@10 和 MRR 分别降 低了 11.2%~16.6% 和 6.7%~15.7%，MAP 降低了 6.6%~18.8%，这突显了双向传播策略在 AST 的节点间捕获上下文信息方面的关键作用，使得程序表示向量中包含更为丰富的结构信息。去除该更新策略可能导致模型无法有效捕捉节点间的关联，从而影响了对缺陷定位的准确性。

对 RQ2 的研究结论如下：首先，联合降噪策略有助于提升缺陷定位的效率；其次，去除自顶向下的更新策略后，BRFN 在所有项目上的各项指标都显著下降。这说明双向传播策略使得程序表示向量中包含更丰富的结构信息，有助于 AST 节点捕获节点间的上下文信息。

表 3 消融实验结果

Tab. 3 Results of ablation experiments

项目	方法	Acc@10	MRR	MAP
Eclipse	BRFN	0.90	0.58	0.49
	BRFN w/o Denoise	0.83	0.53	0.42
	BRFN w/o Top-down	0.75	0.49	0.43
SWT	BRFN	0.91	0.51	0.42
	BRFN w/o Denoise	0.82	0.46	0.40
	BRFN w/o Top-down	0.78	0.43	0.37
Tomcat	BRFN	0.89	0.59	0.45
	BRFN w/o Denoise	0.87	0.57	0.40
	BRFN w/o Top-down	0.79	0.55	0.42
AspectJ	BRFN	0.92	0.57	0.53
	BRFN w/o Denoise	0.89	0.54	0.50
	BRFN w/o Top-down	0.80	0.48	0.46

4 结论

针对现有的缺陷定位方法中缺陷报告的噪音影响及程序代码表示过程中丢失上下文结构信息的问题，本文提出了一种自动化缺陷定位方法 BRFN，通过 TextCNN 和注意力机制对缺陷报告进行特征提取，以便能更准确地理解报告中与实际缺陷相关的内容。此外，利用双向信息传播机制，BRFN 能够更好地捕捉源代码的语法结构和上下文信息，弥补传统的缺陷定位方法在代码表示过程中丢失结构信息的问题。这种融合了自然语言处理和代码分析技术的综合方法提高了缺陷定位的精确性和效率。

下一步工作是引入诸如 CodeBERT 等预训练模型以改变词向量编码，拓展 BRFN 的能力。改进旨在深化对缺陷报告和源代码的语境理解，提高 BRFN 的代码表征质量，增强其灵活性与准确性。另外，考虑在源代码的向量表示中整合数据流图或控制流图等图结构特征以提升 BRFN 语义理解能力，

全面分析源代码多层次结构,进一步提高缺陷定位准确性。这些探索将有助于 BRFN 方法在未来研究和实际应用中发挥更大潜力。

参考文献:

[1] 鞠小林, 姜淑娟, 张艳梅, 等. 软件故障定位技术进展[J]. 计算机科学与探索, 2012, 6(6):481-494.
JU X L, JIANG S J, ZHANG Y M, et al. Advances in fault localization techniques[J]. Journal of Frontiers of Computer Science & Technology, 2012, 6(6):481-494. (in Chinese)

[2] ZHOU J, ZHANG H Y, LO D. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports[C]// Proceedings of the 34th International Conference on Software Engineering (ICSE), June 2-9, 2012, Zurich, Switzerland. New York:IEEE Xplore, 2012:14-24.

[3] YE X, BUNESCU R, LIU C. Learning to rank relevant files for bug reports using domain knowledge[C]//Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, November 16-21, 2014, Hong Kong, China. New York:Association for Computing Machinery, 2014:689-699.

[4] HUO X, LI M, ZHOU Z H. Learning unified features from natural and programming languages for locating buggy source code[C]//Proceedings of the 25th International Joint Conference on Artificial Intelligence, July 9-15, 2016, New York, USA. New York:Association for Computing Machinery, 2016:1606-1612.

[5] LAM A N, NGUYEN A T, NGUYEN H A, et al. Bug localization with combination of deep learning and information retrieval[C]//Proceedings of the 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC), May 22-23, 2017, Buenos Aires, Argentina. New York: IEEE Xplore, 2017:218-229.

[6] LIANG H L, SUN L, WANG M L, et al. Deep learning with customized abstract syntax tree for bug localization[J]. IEEE Access, 2019, 7:116309-116320.

[7] Bug 125101-StackOverFlow in ReferenceType. isAssignableFrom()[CP/OL]. (2006-01-31)[2023-12-01]. https://bugs.eclipse.org/bugs/show_bug.cgi?id=125101.

[8] MOU L L, LI G, ZHANG L, et al. Convolutional neural networks over tree structures for programming language processing[C]//Proceedings of the 13th AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona. New York:Association for Computing Machinery, 2016:1287-1293.

[9] TAI K S, SOCHER R, MANNING C D. Improved semantic representations from tree-structured long short-term memory networks[C]// Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China. Beijing:Association for Computational Linguistics, 2015:1556-1566.

[10] WANG W H, ZHANG K C, LI G, et al. Learning program representations with a tree-structured transformer[C]// Proceedings of the 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), March 21-24, 2023, Taipa, Macao, China. New York:IEEE Xplore, 2023:248-259.

[11] ZHANG B, XIONG D Y, SU J S. Neural machine translation with deep attention[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42(1):154-163.

[12] SOYDANER D. Attention mechanism in neural networks: where it comes and where it goes[J]. Neural Computing and Applications, 2022, 34(16):13371-13385.

[13] ZOHOURIANSHAHZADI Z, KALITA J K. Neural attention for image captioning:review of outstanding methods[J]. Artificial Intelligence Review, 2022, 55(5):3833-3862.

[14] DUBEY S, OLIMOV F, RAFIQUE M A, et al. Label-attention transformer with geometrically coherent objects for image captioning[J]. Information Sciences, 2023, 623:812-831.

[15] CHEN H Y, LI C, LI X Y, et al. IL-MCAM:an interactive learning and multi-channel attention mechanism-based weakly supervised colorectal histopathology image classification approach[J]. Computers in Biology and Medicine, 2022, 143:105265.

[16] CAI W W, NING X, ZHOU G X, et al. A novel hyperspectral image classification model using Bole convolution with three-direction attention mechanism:small sample and unbalanced learning[J]. IEEE Transactions on Geoscience and Remote Sensing, 2023, 61:5500917.

[17] LIU S, LI Y T, FU W N. Human-centered attention-aware networks for action recognition[J]. International Journal of Intelligent Systems, 2022, 37(12):10968-10987.

[18] NIU Z Y, ZHONG G Q, YU H. A review on the attention mechanism of deep learning[J]. Neurocomputing, 2021, 452:48-62.

[19] XIAO X, PAN Y Q, ZHANG B, et al. ALBFL:a novel neural ranking model for software fault localization via combining static and dynamic features[J]. Information and Software Technology, 2021, 139:106653.

[20] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017), December 4–9, 2017, Long Beach, CA, USA. New York:Curran Associates Inc, 2017:6000–6010.

[21] WANG Y L, WANG Y. Feature extraction by using attention mechanism in text classification[C]//Proceedings of the 6th International Conference of Pioneering Computer Scientists, Engineers and Educators (ICPCSEE 2020), September 18–21, 2020, Taiyuan, China. Singapore:Springer, 2020:77–89.

[22] YU Y, SI X S, HU C H, et al. A review of recurrent neural networks:LSTM cells and network architectures[J]. Neural Computation, 2019, 31(7):1235–1270.

[23] ZHANG T H, DU Q F, XU J C, et al. Software defect prediction and localization with attention-based models and ensemble learning[C]//Proceedings of the 2020 27th Asia-Pacific Software Engineering Conference (APSEC), December 1–4, 2020, Singapore, Singapore. New York:IEEE Xplore, 2020:81–90.

[24] HAN J X, HUANG C, SUN S Q, et al. bjXnet:an improved bug localization model based on code property graph and attention mechanism[J]. Automated Software Engineering, 2023, 30(1):12.

[25] WAN Y, SHU J D, SUI Y L, et al. Multi-modal attention network learning for semantic source code retrieval[C]//Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), November 11–15, 2019, San Diego, CA, USA. New York:IEEE Xplore, 2019:13–25.

[26] WANG R Y, ZHANG H W, LU G L, et al. Fret:functional reinforced transformer with BERT for code summarization[J]. IEEE Access, 2020, 8:135591–135604.

[27] 马骏驰，迪骁鑫，段宗涛，等. 程序表示学习综述[J]. 浙江大学学报(工学版)，2023，57(1)：155–169.
MA J C, DI X X, DUAN Z T, et al. Survey on program representation learning[J]. Journal of Zhejiang University (Engineering Science), 2023, 57(1):155–169. (in Chinese)

[28] NGUYEN T D, NGUYEN A T, NGUYEN T N. Mapping API elements for code migration with vector representations [C]//Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), May 14–22, 2016, Austin, TX, USA. New York:IEEE Xplore, 2016:756–758.

[29] XIAO Y, KEUNG J, BENNIN K E, et al. Improving bug localization with word embedding and enhanced convolutional neural networks[J]. Information and Software Technology, 2019, 105:17–29.

[30] CHURCH K W. Word2Vec[J]. Natural Language Engineering, 2017, 23(1):155–162.

[31] ACHEAMPONG F A, NUNOO-MENSAH H, CHEN W Y. Transformer models for text-based emotion detection:a review of BERT-based approaches[J]. Artificial Intelligence Review, 2021, 54(8):5789–5829.

[32] LOU Y L, ZHU Q H, DONG J H, et al. Boosting coverage-based fault localization via graph-based representation learning[C]//Proceedings of the 29th ACM Joint Meeting on European Software Engi neering Conference and Symposium on the Foundations of Software Engineering, August 23–28, 2021, Athens, Greece. New York:Association for Computing Machinery, 2021:664–676.

[33] TIAN Z Z, TIAN B H, LV J J. Combining AST segmentation and deep semantic extraction for function level vulnerability detection[C]//Proceedings of the International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), July 30-August 1, 2022, Fuzhou, China. Cham:Springer, 2023:93–100.

[34] KIM Y. Convolutional neural networks for sentence classification[C]//Proceedings of the 2014 Conference on Empirical Methods in Nature Language Processing (EMNLP), Doha, Qatar. Doha:Association for Computational Linguistics, 2014:1746–1751.

[35] KONDARK G. N-gram similarity and distance[C]//Proceedings of the 12th International Conference on String Processing and Information Retrieval (SPIRE 2005), November 2–4, 2005, Buenos Aires, Argentina. Heidelberg:Springer, 2005:115–126.

[36] HUO X, LI M. Enhancing the unified features to locate buggy files by exploiting the sequential nature of source code[C]//Proceedings of the 26th International Joint Conference on Artificial Intelligence, August 19–25, 2017, Melbourne, Australia. New York:Association for Computing Machinery, 2017:1909–1915.

(责任编辑：仇慧)