计算机科学
COMPUTER SCIENCE

面向软件缺陷报告的缺陷定位方法研究与进展

倪珍，李斌，孙小兵，李必信，朱程

引用本文

倪珍，李斌，孙小兵，李必信，朱程. 面向软件缺陷报告的缺陷定位方法研究与进展[J]. 计算机科学, 2022, 49(11): 8-23.

NI Zhen, LI Bin, SUN Xiao-bing, LI Bi-xin, ZHU Cheng. Research and Progress on Bug Report-oriented Bug Localization Techniques[J]. Computer Science, 2022, 49(11): 8-23.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220200117

面向软件缺陷报告的缺陷定位方法研究与进展

倪 珍[1] 李 斌[1] 孙小兵[1,2] 李必信[3] 朱 程[1]
1 扬州大学信息工程学院 江苏 扬州 225127
2 南京大学计算机软件新技术国家重点实验室 南京 210023
3 东南大学计算机软科学与工程学院 南京 211189
(nizhen@yzu.edu.cn)

摘 要 软件缺陷定位是软件缺陷修复任务的一个重要步骤。面向软件缺陷报告的缺陷定位方法以描述缺陷产生现象的软件缺陷报告作为查询,以项目的源代码作为语料库,通过分析缺陷报告与源代码单元之间的相关关系,设计缺陷报告与源代码单元之间相关度的计算方法;随后,挖掘各类软件历史仓库来创建缺陷定位数据集,构建缺陷定位模型,以识别缺陷报告所描述的缺陷对应的源代码单元(即缺陷位置),实现缺陷定位。对近年来国内外学者在该研究领域取得的成果进行了系统总结。首先,介绍了软件缺陷定位的相关概念,归纳了面向软件缺陷报告的缺陷定位方法的主要流程;其次,围绕定位流程中的3个关键步骤梳理了已有研究工作;然后,总结了缺陷定位领域常用的实验数据集和实验评估指标;最后,对未来研究可能面临的挑战进行了展望。

关键词:软件缺陷定位;软件缺陷报告;定位模型;信息检索;深度学习
中图法分类号 TP311

Research and Progress on Bug Report-oriented Bug Localization Techniques

NI Zhen[1], LI Bin[1], SUN Xiao-bing[1,2], LI Bi-xin[3] and ZHU Cheng[1]
1 School of Information Engineering, Yangzhou University, Yangzhou, Jiangsu 225127, China
2 State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
3 School of Computer Science and Engineering, Southeast University, Nanjing 211189, China

Abstract Software bug localization is an important task in bug fixing process. The bug report-oriented localization approaches typically use software bug reports that describe the phenomenon of bugs as queries, and source code as corpus. First, correlations between the bug report and each source code unit are analyzed. Then, a bug localization data set is created by mining the software repository, aiming to construct a bug localization model to localize the source code unit (i.e. bug location) corresponding to the bug report. This paper offers a systematic survey of existing research achievements of the domestic and foreign studies of bug localization in recent years. First, the related concepts in bug report-oriented bug localization are introduced, and the main localization process is summarized, followed by discussing the existing research works that focus on the three key steps in the localization process. Then, the commonly used data sets and evaluation metrics for bug localization are summarized. Finally, future work in this research area is discussed.

Keywords Software bug localization, Software bug report, Localization model, Information retrieval, Deep learning

1 引言

软件缺陷定位(Software Bug Localization, SBL)指找出导致软件某些功能失败或执行异常或不符合规范的缺陷位置的过程[1]。需要说明的是,本文讨论的面向缺陷报告的定位方法不同于程序故障定位[2](需运行测试套件),而是静态地分析缺陷报告和相关项目的源代码以找到导致缺陷的源代码单元。由于缺陷数量和代码规模不断增长,如何自动定位缺陷位置以提高软件缺陷修复效率与质量,成为研究人员近年来关注的热点之一。

近年来,一些研究者利用信息检索技术,使用文本检索模型计算缺陷报告与待定位源代码单元之间的相关度,并据此

到稿日期:2022-02-21 返修日期:2022-05-12
基金项目:国家自然科学基金(61972335,61872312);南京大学计算机软件新技术国家重点实验室开放课题(KFKT2020B16)
This work was supported by the National Natural Science Foundation of China(61972335,61872312) and Open Funds of State Key Laboratory for Novel Software Technology(Nanjing University)(KFKT2020B16).
通信作者:李斌(lb@yzu.edu.cn)

来对源代码单元进行排名，将排名靠前的源文件返回给开发人员。然而，传统的信息检索模型大多无法有效弥合编程语言和自然语言之间的词汇差异[3]。因此，近年来的研究已经开始探索智能计算技术的潜力，如机器学习、深度学习和进化算法等。其中基于深度学习的缺陷定位模型是近年来的主流方法[4-32]。此外，研究者开始关注使用其他来源的数据特征以优化定位模型的输出和重构定位模型的输入，以辅助提高现有缺陷定位方法的准确性。

已有文献对基于信息检索的软件缺陷定位（Information Retrieval Based Bug Localization，IRBL）的研究进展进行了详细的综述[33-36]，这些研究对 IRBL 进行了详细的分析与汇总，侧重点各有不同。文献[33]按照文本挖掘技术的不同简要总结了现有的 SBL 模型。文献[34]主要从数据源和技术评估指标两个方面分析总结了现有的基于信息检索的缺陷定位研究工作进展。文献[35]总体上围绕基于信息检索的定位方法（IRBL）从改良 IRBL 方法和评估 IRBL 方法两个层面展开综述，其中，改良 IRBL 从更换 IR 模型、使用数据特征分析、进行查询重构和应用深度学习 4 个方面详细介绍了近年来的研究进展，在进行查询重构方面主要介绍了对缺陷报告进行查询重构。文献[36]主要从数据源、检索模型、场景应用 3 个角度介绍了基于信息检索的缺陷定位研究进展。

上述综述收集了 2000－2019 年的相关文献，本文检索的是 2000－2020 年的相关文献。在内容上，与上述综述工作不同的是，本文对已有工作的研究进展的综述是按照定位流程中对不同关键步骤的优化来分类阐述的。鉴于已有综述已对 IRBL 研究做了详细、全面的分析总结，本文仅简要总结了此类方法的进展，而重点介绍和分析基于 DL 的定位方法。

具体地，本文将本领域的已有工作分为定位模型的研究、对模型输出的优化研究和对模型输入的优化研究 3 个方面。首先，在定位模型方面，本文总结了各类定位模型如何表示、抽取查询和语料库的特征来计算相关度以达到定位目的。其中，本文重点梳理了基于 DL 的自动抽取文本语义特征的定位方法，并对此类定位模型进行了对比分析和总结。其次，在模型输出的优化方面，本文介绍了其他辅助定位的特征，并总结了现有方法如何使用这些特征来优化基础定位模型输出的相关度得分。最后，在模型输入的优化研究方面，本文从查询优化和语料库优化两方面介绍了已有工作如何对缺陷报告和源代码的表示进行优化，其中包含了利用查询重构方法提升定位效果。同时，本文对以上 3 个方面的研究现状进行了总结和纵横向的优缺点分析。另外，本文根据实证研究总结了现有的面向软件缺陷报告的自动软件缺陷定位方法的数据集和评估标准。更多详细信息也可参考我们的综述网址[1)]。

2　文献选取与统计

文献选取标准：本文选取的文献是针对面向软件缺陷报告的软件缺陷定位方法以及技术评估等方面提出的方法或技术，或者说选取的文献是为面向软件缺陷报告的基于静态分析的软件缺陷定位方法的相关理论技术提供实证研究。

文献检索和筛选步骤：

（1）检索数据库：IEEE，ACM，DBLP，Springer，Google scholar 以及中国知网（CNKI）等论文数据库。

（2）检索关键词：包括“缺陷定位”“面向缺陷报告”“基于 IR 的缺陷定位”“基于 DL 的缺陷定位”“静态缺陷定位”的中英文关键词；检索时间范围为 2000 年 1 月 1 日至 2020 年 12 月 31 日。

（3）文献检索与筛选：使用以上关键词在数据库中检索，对检索出来的文献集合，人工查看文章标题、关键词、摘要，以及粗略浏览文章内容，去除“未使用缺陷报告”“主要使用动态的故障定位技术”“主要基于测试例执行”“主要基于程序频谱”“概念/特征/关注识别”的定位技术或实证研究的文献；最后查看文献的参考文献，筛查出相关文献。

基于以上步骤，本文最终汇总了 142 篇面向软件缺陷报告的缺陷定位方法的相关文献，如图 1 所示，此领域的文献数量随时间推移呈现波动上升的趋势且在近两年达到最高，说明面向软件缺陷报告的缺陷定位研究愈来愈受到关注，并且是近两年的研究热点。自 2015 年以来，深度学习技术在缺陷定位方面的应用较为火热。自 2019 年起，每年基于深度学习的缺陷定位文献数量均超过了基于信息检索的缺陷定位文献数。从主要使用的关键技术来看，主要使用 IR 技术的文献[37-95]占 42%，主要使用 DL 方法的文献[4-32]（包含 DL 与 IR 结合的方法）占 20%，主要使用查询重构技术的文献[96-109]占 10%，实证研究、综述、工具等其他类的文献[33-36,110-145]占 28%。

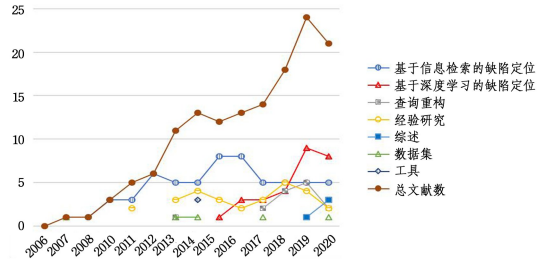

图 1　面向软件缺陷报告的缺陷定位方法的文献信息汇总

Fig. 1　Literature statistics of bug report-oriented bug localization

3　缺陷定位基本概念

3.1 定义

缺陷定位指由开发人员、测试人员或用户提交文档形式的缺陷报告，它描述了所报告的缺陷。被指派去修复缺陷的开发人员将分析此报告，并在程序代码中搜索可疑代码。此过程被称为缺陷定位[94]。具体地，定位输入包括缺陷报告和候选的源代码单元，定位的目标是识别出为修复该报告所需修改的源代码单元，定位任务的输出是对输入源代码单元按可疑度由高到低排序的列表。

3.2　步骤

图 2 给出了面向缺陷报告的缺陷定位的通用步骤。

1) https://buglocalization.gitee.io/

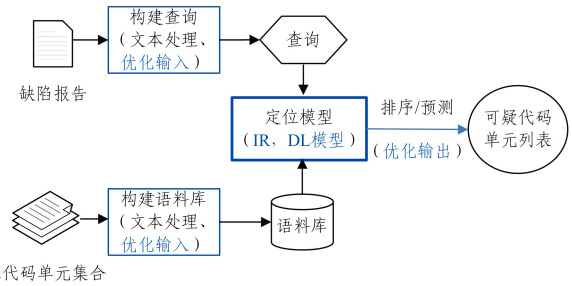

图 2 面向软件缺陷报告的缺陷定位的通用步骤

Fig. 2 General procedures of software bug report-oriented bug localization

(1)查询构造：一般从缺陷报告的文本中构建查询，并使用其在后续已建立索引的源代码语料库中搜索相关源代码单元。一般从缺陷报告的标题和描述中提取词语并进行预处理(如词法分析、去停用词、词干化等)，最后形成查询。

(2)语料库创建：构建待定位的源代码语料库，从相关的项目所有源代码中抽取源代码单元(如文件、方法或语句)并进行预处理。现有研究大多以文件为单位提取源代码，创建语料库后，通常还要对语料库中的所有源代码单元进行索引，以便后续检索和排序。

(3)定位模型：通过抽取已创建查询和语料库中的数据特征，根据特征计算查询与语料库中每个源代码单元的相关度得分，以对待定位的源代码单元进行排名，最后根据此相关度得分给出对应的可疑源代码单元的排名列表。

表 1 列出了面向软件缺陷报告的定位方法中代表性的定位技术的名称、定位级别、使用的模型以及数据分析特征。

表 1 面向软件缺陷报告的代表性定位方法综合汇总

Table 1 Comprehensive statistics of representative bug report-oriented bug localization methods

年份	方法名称	定位粒度	IR/DL 模型	数据分析特征				
				版本历史	相似报告	代码结构	堆栈踪迹	其他
2010	Lukins[37,49]*	M	LDA					
	Nichols[38] *	M	LSI					
2011	LSICG[71,92]	M	LSI					依赖关系
2012	TFIDF-DHbPd[59]	F	DFR	√				
	BugLocator[94]	F	rVSM		√			
2013	BL UiR[62]	F	Indri		√	√		
	Tantithamthavorn[63-64] *	F	VSM	√				
	Kim[13] *	F	word embedding					
2014	VSMcomposite[40]	F	VSM	√	√	√		
	Lobster[41]	Cl	VSM			√	√	
	BRTracer[65]	F	rVSM		√		√	代码文本分段
	AmaLgam[66]	F	Mixed IR	√	√	√		
2015	LDACG[43]	M	LDA					依赖关系
	MrVSM[67]	F	rVSM	√		√		
	BLIA[68-69]	F	rVSM	√	√	√	√	代码文本分段
	Tong[76] *	F	VSM				√	词性
	Hyloc[26]	F	DNN	√				API 文档
2016	BugCatcher[48]	F	Lucene			√		
	AmaLgam+[74]	F	Mixed IR	√	√	√	√	报告者信息
	Locus[75]	F/Ch	VSM	√		√		
	Ye[4]	F	word embedding					API 文档、教程
	NP-CNN[28]	F	CNN					
	DrewBL/CombBL[14]	F	word embedding, VSM	√				
2017	L2SS+[52]	F	LLDA					元数据
	DNNLOC[27]	F	DNN, rVSM	√				
	LS-CNN[29]	F	CNN, LSTM					
	DeepLocator[31]	F	word embedding, CNN	√		√		
2018	STMLOCATOR[54]	F	LLDA	√			√	文件大小
	Kyaw[79] *	F	VSM		√	√	√	结构级别
	CNN_Forest[15]	F	word embedding, CNN			√		
	BugTranslator[6]	F	RNN, LSTM		√	√		
	Xiao[7] *	F	CNN, RNN, LSTM			√		
	Loyola[17] *	Ch	LSTM					依赖关系
2019	DeepLoc[32]	F	word embedding, CNN	√				
	D&C[87]	F	Mixed IR			√	√	代码文本分段
	SCOR[5]	F	word embedding					
	TRANP-CNN[30]	F	word embedding, CNN					
	CAST[8]	F	CNN			√		依赖关系
	FineLocator[100]	M	word embedding	√		√		依赖关系
	LSLS[12]	F	word embedding, VSM			√		
2020	CG-CNN[19]	F	CNN, LSTM			√		
	MD-CNN[22]	F	CNN	√	√	√		API 文档
	Scaffle[23]	F	CNN, VSM					崩溃跟踪
	CooBa[20]	F	CNN, GCN, LSTM			√		
	STMLOCATOR+[53]	F	LLDA	√			√	文件大小，元数据

注：√ 表示该方法用到了对应的数据分析特征；* 表示没有专用定位名称的使用作者名；定位粒度缩写 F：File(文件)，Cl：Class(类)，M：Method(方法)，Ch：Change(代码变更)

4　面向缺陷报告的软件缺陷定位方法

本节从定位模型、模型输出优化和模型输入优化3个方面来梳理面向缺陷报告的定位方法的研究进展。

4.1　定位模型的分类

4.1.1　基于信息检索的定位模型

基于信息检索的定位方法一般使用传统的IR或ML技术，如LSA(Latent Semantic Analysis)[146]，LDA(Latent Dirichlet Allocation)[49]，VSM(Vector Space Model)[40]等。这类方法通常先人工定义文本词汇特征(如TF-IDF[94])来表示查询和语料库，然后计算两者之间的文本词汇相似度来排序代码单元。

文献[38]首次利用历史的缺陷报告对LSI(Latent Semantic Indexing)模型进行了扩展，并提出利用有关源代码中实体的可用附加信息可以进一步帮助信息检索模型识别关系。为了评估LSI中参数对缺陷定位的影响，文献[39]通过实验证明了LSI中词汇文档矩阵的加权函数和语料库的过滤方法会对基于LSI的缺陷定位的性能产生重要影响。Lukins等[37,49]提出了基于LDA的缺陷定位方法，该方法与基于LSI的定位方法相比具有模块化和可扩展的优势。不同于前述基于LSI或LDA的定位方法只考虑待定位缺陷的报告和代码的文本相似性，文献[94]提出的BugLocator使用修订的向量空间模型(rVSM)，根据初始缺陷报告和源代码之间的文本相似性对所有文件进行排名，同时考虑了相关的已修复的相似缺陷报告的信息。为了提升定位准确率，文献[40]提出组合多个VSM变体以改善缺陷定位的效果，文献[63]使用概率检索模型从缺陷报告中抽取特征向量，文献[57]提出可以使用针对对象之间相似性的不同维度的不同IR方法来增强彼此结果的可信度。

在经典信息检索模型中，VSM与LSI，LDA相比有更好的性能表现。VSM模型是所有经典模型中被使用次数最多的模型。IR模型配置的选择对缺陷定位性能影响较大，需要针对不同数据集做不同选择[114,120,144]。

4.1.2　基于深度学习的定位模型

基于深度学习的定位方法(Deep Learning Based Bug Localization，DLBL)通常将缺陷定位问题看作是分类问题(源代码单元与缺陷报告是否相关)。一般通过训练语言模型对输入数据(预处理好的已标记的查询和语料)进行向量化表示[147]，在训练此向量化表示的过程中即自动提取了输入的语义特征(如上下文特征)，然后再计算两个向量之间的相似度[4-5,12,14,16]。近年来，基于深度学习的方法常使用神经网络(如DNN，CNN，RNN等)作为定位模型[7-9,26-31]，使用独立的神经网络分别自动提取缺陷报告和源代码的特征，训练之后各神经网络的最后一个隐层输出即为原始输入的特征向量的表示。之后通常接上一个全连接层来融合上一步提取的缺陷报告和代码的特征。最后接上一个分类器，分类器的输出结果即为查询与语料的相关性分类结果(一般是概率值，表示查询与语料相关的可能性)。此结果即为代码单元和待定位缺陷之间的文本语义相似度。本文收集的29篇主要使用深度学习模型的定位方法中，有3篇[9,13,16]主要使用词嵌入模型，有9篇[8,10,15,18,22,28-31]主要使用CNN模型，有3篇[6,11,17]主要使用RNN/LSTM模型，有5篇[7,19-21,32]综合使用了多个DL模型，其余9篇[4,5,12,14,23-27]在已有IR模型的基础上结合使用了DL模型改进定位效果。

(1)词嵌入(Word Embedding)

神经网络语言模型(Neural Network Language Model，NNLM)[148]使用低维向量表示每个单词，这些向量被称为“词嵌入”[147]。对于缺陷定位任务，基本方法是将已定位好的历史缺陷报告和对应的源代码单元作为训练用的输入，使用语言模型学习自然语言单词和代码词的嵌入，使含义相似的词与相似的向量嵌入关联，然后通过计算待定位的缺陷报告的所有向量和源代码单元的所有向量之间的相似度来判断待定位报告和各源代码单元是否相关。

考虑到领域文档和源代码文本之间的语义相似性，文献[4,16]使用Skip-gram模型在API文档、教程和参考文档上训练词嵌入。实验结果表明，从领域文档训练词嵌入可以使语义相近的自然语言词汇和编程语言token在向量表示上更接近，从而缓解词汇差异的问题。

词嵌入能从缺陷报告和源代码中学习上下文相关的语义信息，并快速有效地计算它们之间的相似度。但是，如果某些单词对从未出现在相同的上下文中而且彼此相关，则词嵌入将它们分配为接近向量的可能性较低，这会限制词嵌入的性能。此外，因为词嵌入的重点是单词而不是句子，很难学习单词之间的相对位置，即词嵌入很难学习到句子的语义，导致无法学习到缺陷报告与源代码单元两个整体之间的相关性特征。因此，词嵌入对缺陷定位准确率的提升空间仍然有限。

(2)卷积神经网络(Convolutional Neural Network，CNN)

在自然语言处理中，CNN[149]也被应用于学习单词向量表示[150]。已有研究[150]证明，即使具有较少超参数的简单CNN也能表现出很好的文本分类性能，且CNN已被广泛应用于学习句子极性的研究中。已有研究[151-153]表明，深度学习适用于解决一些软件工程问题，其中CNN也可用于学习编程语言的语义特征[153]。因此研究者们考虑将CNN应用于定位任务，并研究通过学习来关联缺陷报告中与领域相关的术语和源代码中的不同token，最终正确分类缺陷报告和源代码的文本。

通常地，基于CNN的定位方法的目标是使用已修复的缺陷报告和其对应的相关源代码单元训练一个预测模型，训练好的模型能够预测待定位缺陷报告和每个源代码单元之间的相关度值，相关度值越高，对应的源代码单元是错误的源代码单元的可能性就越高。CNN模块主要用于自动提取缺陷报告或者源代码的特征。作为查询的缺陷报告是用自然语言描述的，因此，基于CNN的定位方法一般采用和自然语言处理领域相同的方法来表示缺陷报告文本。而源代码由编程语言描述，文本的语法结构和自然语言不同，因此应用CNN对源代码文本进行表示时会有所不同。学习源代码的结构语义是CNN应用于定位任务的最大优点。不同层次的卷积和池化操作使CNN能够以分层方式识别层次结构数据的特征，而用编程语言描述的源代码正是这种层次结构的数据，因此CNN非常适用于抽取源代码的结构特征。

文献[28]中 NP-CNN 主要由两个连续部分组成:第一部分是语言内特征提取层,它分别使用缺陷报告和源文件提取基于多层卷积神经元的特征,其中源代码的卷积操作专门用于反映程序结构;第二部分是跨语言特征融合层,它将来自缺陷报告和源文件的提取特征组合成统一表示。除了使用具有多个卷积核的 CNN 分别从缺陷报告和源代码衍生的词向量中提取语义和结构特征之外,文献[15]还使用级联的随机林集合进一步提取更深的特征,并观察缺陷报告和源文件之间的相关关系。该经验研究的结果表明,缺陷报告和源文件中的语义和结构信息对改善缺陷定位至关重要。

从源代码(语料库)的特征提取的角度来看,CNN 对源代码的卷积操作可反映程序的结构,即最终的特征表示中蕴含了源代码特定的结构语义信息,可缓解前述的词汇不匹配问题。但目前已有的基于 CNN 的定位研究工作都是在较小规模的数据集上实现的,且已有实证研究[126]证明 CNN 在大型数据集上定位比在小型数据集上定位消耗的时间和计算资源更多。这是因为当源代码文件长度增加时,CNN 的层数也会增加,CNN 网络需更新的参数可能会以指数形式增长,模型计算的耗时也就更长。

(3)循环神经网络(Recurrent Neural Network,RNN)和长短期记忆网络(Long Short-Term Memory,LSTM)

RNN[154]是一类用于处理序列数据的神经网络模型。LSTM 网络[155]是 RNN 的一种变体,在一定程度上解决了 RNN 的梯度消失的问题。

通常,基于 RNN 的缺陷定位方法使用已修复的缺陷报告和其对应的源代码单元训练一个翻译模型,训练好的模型能将待定位的缺陷报告翻译成源代码 token 的序列,最后计算此 token 序列和各个待定位的源代码单元之间的相似度。相似度值越大,对应的源代码单元是错误的源代码单元的可能性就越高大。此类模型主要由具有 LSTM 的基于注意力的 RNN 编码器/解码器组成。一个 RNN 编码器将自然语言的源语句(缺陷报告)编码为多个上下文向量,然后另一个 RNN 解码器将上下文向量解码为缺陷源文件的代码 tokens 序列。其中,RNN 编码器学习到的上下文向量是关联源句子和目标句子之间的桥梁。对于定位任务,即将缺陷报告翻译成代码序列之后,最后计算翻译出来的代码序列和待定位的每个代码序列之间的相似度。

文献[6]提出的缺陷定位模型 BugTranslator 使用一个 RNN 并借助领域文档将缺陷报告编码为多个上下文向量,然后由另一个 RNN 解码为缺陷源文件的代码 tokens。此方法利用了从缺陷报告与源文件中提取的语义信息之间的相关性,缓解了词汇鸿沟问题,提高了缺陷定位的效率。文献[17]提出了一种代码更改级别的定位方法。该方法从语法和代码更改依赖关系的角度,使用基于注意力机制的 LSTM 模型,从项目历史记录中提取的源代码更改中学习特征表示。由于每个更改都与源代码文件相关联,因此更改级别的定位方法不仅为开发人员提供了与报告相关的更精确的缺陷位置,而且能更方便地识别导致该缺陷的代码更改,从而简化了后续的修复任务。

从缺陷报告和源代码特征提取的角度来看,基于 RNN 的定位模型利用具有存储能力的 RNN 来学习单词之间的相对位置,且使用 LSTM 能捕捉时序间隔较大但在整个句子中语义相似的词汇或代码 tokens 之间的相关关系,从而获取缺陷报告和源代码的语义相关性特征,也可缓解前述的词汇不匹配问题。虽然 RNN 能很好地处理文本数据变长并且有序的输入序列,但是源代码文本的语法结构并不是线性的顺序结构,而目前少量的基于 RNN 的定位方法都将源代码文件视为和缺陷报告一样用自然语言描述的线性文本,忽略了源代码的非线性的结构语义信息。此外,LSTM 能在一定程度上缓解长依赖问题,但序列长度超过一定限度后,梯度依然会消失。因此,基于 RNN 的定位模型对定位效率的提升效果有待进一步的研究。

最近,有少数研究工作[7,19-21,32]也关注综合各种神经网络模型的定位方法。为了解决跨项目定位方法难以捕获特定于单独项目的特征的问题,文献[20]提出了一种对抗的迁移学习缺陷定位方法 CooBA,结合 LSTM、CNN、GCN 和对抗性学习以确保多个项目之间共享的公共特征和项目特有特征能被有效提取。

4.1.3 深度学习与信息检索技术结合的定位方法

目前已有的组合技术的定位研究[5,12,14,23-27]大多分别使用基于 IR 和基于 DL 的定位模型提取文本词汇相似度和文本语义相似度,然后对两个相似度值进行加权求和,其中权值采用人工调整或学习的方式获得,最后根据组合的相似度得分来排序待定位的源代码单元。上述研究结果表明,两种相似度在定位任务中可产生互补作用,在一定程度上能提升定位的效率。但从实验结果的分析(见表 2)来看,目前这种组合的策略对定位任务性能的提升效果并不明显,且目前还没有研究工作对这种组合技术的定位模型进行实证分析,因此如何结合两种定位模型来提高定位效率仍然值得探索。

4.1.4 小结

本小节以定位模型使用的两个关键技术(信息检索和深度学习)为侧重点介绍并分析了面向缺陷报告的软件缺陷定位的研究进展。IR 类的技术是本领域主流的定位模型,其中 VSM 模型被使用的频次最高。深度学习的 CNN、词嵌入、RNN 也有一定的应用。基于 IR 的定位模型首先人工定义查询和语料库的相似度特征,然后采用文本检索或机器学习技术来抽取特征,最后根据抽取的特征计算查询和语料的相似度,以对语料库中相应的待定位源代码单元进行排序。已有的研究工作表明,基于 IR 的定位模型(尤其是使用 VSM 的定位模型)取得了较好的定位效果,并且 IR 模型在实验中所需要的计算资源较少,利用相应的工具能进行快速定位。因词汇鸿沟问题,基于 IR 的定位方法的定位性能的提升空间有限,于是研究者近年来开始考虑利用基于 DL 的定位模型来缓解词汇鸿沟问题。基于 DL 的定位模型无须事先人工定义查询和语料库的相似度特征,一般通过训练神经网络来自动表示或抽取数据语义特征,最后分类器根据语义相似度来预测代码单元对于待定位缺陷的相关度得分。由于神经网络的语言模型能够学习自然语言和编程语言的语义特征,因此基于 DL 的定位方法通过关联语义相关的缺陷报告和源代码来有效缓解词汇鸿沟问题,但 DL 模型的实用性还存在局限性。

词嵌入利用上下文信息能使语义相近的词汇在表示值上也接近，但因过度依赖上下文而使其提升定位效率的能力有限。CNN 在结构语义的学习上表现出强大优势，但面对大规模数据集会因消耗资源过多而变得不实用。RNN 能较好地学习句子级别的语义特征，但对非线性文本的结构语义学习能力较弱。因此，如何合理利用和改进这两类模型以提高定位性能仍然是本领域的研究重点。

笔者从最近的研究工作中按照相同定位粒度（即文件级别）和相同项目（即 JDT）抽取并汇总了几个在实验上表现较好的代表性定位模型的实验结果（见表 2）。从表 2 可以看出，使用 DL 类模型的定位方法整体性能优于使用 IR 类的定位模型，这可能得益于前述 DL 模型能够学习语义特征，而使用 IR 和 DL 混合技术的定位方法并未明显提升基于 DL 的定位模型的性能。因此如何有效地结合两类模型来充分提高定位方法的性能是本领域的一个重要挑战。在 DL 模型中，CNN 的整体性能略优于其他 DL 模型，原因可能是 CNN 更擅长学习源代码的层次结构特征。在各个类别内部，混合数据特征的定位方法的整体性能优于单纯使用 IR 或 DL 的定位方法，这说明使用除计算缺陷报告和源代码之间相关度之外的其他特征来优化模型的输出可提高定位方法的性能。模型输出优化的方法将在 4.2 节详细介绍。

表 2　面向软件缺陷报告的代表性定位方法的实验结果汇总

Table 2　Experimental results of several representative bug report-oriented bug localization methods

类别	具体模型	研究工作	使用输出优化特征	Top-5	MAP	MRR
IR	rVSM	BugLocator[94]	相似报告	0.400	0.290	0.370
	VSM	Ye[73]	版本历史，相似报告，代码结构，其他	**0.552**	**0.340**	**0.420**
DL	CNN	NP-CNN[28]		0.669	0.383	0.461
		CNN Forest[15]			0.423	0.514
		MD-CNN[22]	版本历史，相似报告，代码结构，其他	**0.720**	**0.450**	**0.530**
	RNN(LSTM)	BugTranslator[6]		0.581	0.340	0.410
	CNN＋RNN(LSTM)	Xiao[7]			0.350	0.410
	word embedding＋CNN	DeepLoc[32]	版本历史	0.650	0.440	**0.530**
IR＋DL	word embedding＋VSM	Ye[4]	版本历史，相似报告，代码结构，其他	**0.650**	**0.420**	**0.520**
	DNN＋rVSM	DNNLOC[27]	版本历史	0.650	0.342	0.452

注：表中加粗的数据表示该类别中实验结果较优的定位方法的评估值

4.2　使用相关特征优化模型输出

在考虑更换和改进定位模型本身的同时，研究者们也研究了如何优化定位模型的输出，即使用其他来源的数据特征计算相关度值，结合基本模型输出的相似度值，辅助调整待定位源代码单元的相关度，使缺陷报告对应的真实缺陷源代码单元在列表中的排序更靠前，从而提高定位准确度。

结合不同的数据来源综合计算相关度得分来进行缺陷定位的方法被称为混合数据特征的缺陷定位方法[12-14,26-27,58-59,61-62,65-69,72-75,77-81,84-85,94,100]。目前已有很多研究使用了此方法。表 1 也汇总了本文总结文献所用的优化输出的数据特征。常用的优化输出的数据分析特征主要是版本历史、相似报告、文本结构和堆栈跟踪，其他特征主要包含 API 文档、依赖关系、代码文本分段等。本节将根据这些数据特征的来源介绍对定位模型输出优化的研究进展，以及使用这些特征的原因，最后总结这方面工作的优缺点。

4.2.1　用于输出优化的数据特征及其应用

(1)版本历史

版本历史即源代码的更改历史记录。已有研究[156]表明，源代码的更改历史记录提供了有助于预测容易出错的代码文件的信息。定位中常用的版本历史特征一般是源代码的修改频率和修改新近度。使用版本历史数据特征的直觉是：经常被修复的源代码单元更容易导致缺陷；与很久以前修复或从未修复过的源代码单元相比，最近修复的源代码单元更有可能导致缺陷。

文献[73]使用源代码修复频率和修复新近度的得分辅助提高定位效率。文献[63-64]提出了一种挖掘代码共同变更历史的方法来提升缺陷定位性能。文献[59]利用源代码文件的修改历史和缺陷历史分别计算出源代码文件的修改历史和缺陷历史的先验概率，并利用其调整源代码文件的排序。文献[75]提出了一种使用源代码变更信息来辅助定位软件变更中的缺陷的方法 Locus。文献[83]指出版本问题在源代码中有自己的缺陷模式，并提出了与版本相关的缺陷定位方法。文献[31-32]通过在全连接层中添加缺陷修复的新近度和频率作为代价函数的两个惩罚项来增强常规的 CNN，最终提升缺陷定位的性能。文献[54,66]提出的自动定位方法均考虑了利用版本历史来优化输出。

(2)相似缺陷报告

相似的缺陷报告通常具有相似的相关缺陷源代码文件[94]，因此，对于新缺陷报告，可以检查其类似的已修复的缺陷及其相关文件，以调整最终文件列表的顺序，其中相似缺陷报告的相关文件排名较高。

BugLocator[94]使用了待定位报告和先前已修复报告的相似度来调整代码文件的相关度得分。文献[77]利用以前修复的缺陷报告和新接收的缺陷报告之间的关系，使用文本匹配和多标签分类来提升缺陷定位的性能。

(3)代码结构

代码结构反映了源程序的语义特征，使用代码结构计算相关度得分一般与文本相似性得分一起考虑。

BLUiR[62]使用了每个代码结构类型组合的相关度得分。文献[79]提出结构组件的重要级别是不同的，因此基于在缺陷报告中的潜在重要性来考虑源代码结构字段（类名、方法名、变量名等）的重要级别。LSLS[12]在计算文本词汇相似度和文本语义相似度时，都根据缺陷报告的 POS 标签以及从 AST 提取的源代码文件的方法和类名称来调整模型中词的

权重。文献[48,67]在输出优化中也考虑了代码结构。

(4)堆栈跟踪

有时,缺陷报告包含失败线程的堆栈跟踪信息。堆栈跟踪显示了直到程序异常终止为止所执行的指令序列[157]。它也可以被视为与该缺陷潜在相关的源代码文件的排名列表。使用堆栈跟踪的直觉是,堆栈跟踪中排名靠前的文件更容易包含缺陷,应具有更高的相关度得分。文献[117]也研究验证了堆栈跟踪信息对基于 IR 的缺陷定位方法的影响。

文献[65]在实验中使用了堆栈跟踪的提升分数,它是堆栈中文件排名的倒数。文献[41]结合基于堆栈跟踪的相似性和 IR 技术提供的文本相似性来检索与缺陷报告相关的代码元素。文献[68-69]也提出了一种混合的缺陷定位方法 BLIA,该方法利用了缺陷报告中的文本和堆栈跟踪、源文件的结构化信息和源代码更改历史计算 4 个相关度分数,根据以上 4 个分数的组合,对可疑源代码文件进行排序。

(5)其他

还有研究使用调用依赖关系[43,71,92]、领域文档(API 文档[73]和软件教程[4])、元数据[52,53]、文本词性[72]、报告者信息[74]、软件项目的需求信息[85]、缺陷修复提交消息[81]等其他来源的数据分析特征来辅助提高缺陷定位的准确率。

4.2.2 小结

为了提升真实的缺陷代码单元在定位模型输出的排序列表中的位置,除了使用从缺陷报告的自然语言文本和待定位源代码的文本中抽取的文本相似度特征之外,现有研究还使用了其他来源的数据特征辅助计算源代码单元的相关度值,最终提高缺陷定位的准确性。这些特征来自多种缺陷相关的数据源,包含堆栈跟踪、项目版本历史、代码结构、相似历史缺陷等。从上述已有研究工作的进展来看,由于这种优化方式简单且易扩展,使用混合数据特征优化缺陷定位模型输出的策略已被广泛应用于两类模型(IR 和 DL)的定位研究中,其中基于深度学习的定位方法大多使用版本历史特征和代码结构特征来提高定位效率,这可能是得益于神经网络模型在学习语义特征(如上下文语义和结构语义)上有相对较强的优势。

4.3 优化模型的基本输入

实证研究[121-122]结果表明,如果缺陷报告缺乏丰富的结构化信息,如相关的程序实体名称,那么基于 IR 的缺陷定位技术就不能很好地执行。相反,缺陷报告中过多的结构化信息,如堆栈跟踪,通常可能不利于自动缺陷定位。因此,输入的质量对定位模型也有重要影响。目前已有少数研究工作对定位模型的输入进行重构优化,以辅助提高定位效率。目前定位模型输入的优化方面的研究工作主要在于优化查询(缺陷报告)和优化语料库(待定位的源代码)两个方面。本节将从这两方面梳理并总结现有研究工作。

4.3.1 查询优化

文献[102]详细研究了缺陷报告的质量对基于 IR 的缺陷定位的性能的影响,提出克服这些限制的可能方法,即查询重构(Query Reformulation,QR),并提出了通过上下文感知的查询重新制定来改进基于 IR 的缺陷定位技术[104,107]。通过将适当的关键字选择算法、上下文感知、Stack Overflow 上的众包知识和大规模数据分析整合到查询重构过程中来改进代码搜索。文献[96]提出了基于邻近度的 QR 技术,通过从响应初始查询而检索到的排名最高的代码中提取特定的附加术语来丰富用户的搜索查询。考虑在源代码中处理相同概念的术语通常在相同的文件中彼此接近,这些注入到查询中的附加术语根据位置邻近性被认为是与源代码中的原始查询术语接近的术语。文献[97]提出使用缺陷报告的描述部分来重构查询[106]。实验结果表明,结合缺陷报告标题和观察到的行为来重构查询是最好的策略。文献[108]提出使用附件来扩展缺陷报告,并通过减少其中的噪音项来优化查询。

4.3.2 语料库优化

为了实现细粒度的缺陷定位,文献[100-101]提出了一类方法级细粒度的缺陷定位方法,通过查询扩展相邻方法的元素来增强具有短长度的方法的表示。

现有的源代码文本表示方法中使用的源代码文件的程序抽象语法树(Abstract Syntax Tree,AST)通常具有过多的冗余或不相关的节点,导致训练时间长、维度爆炸、过拟合等问题。文献[8]提出使用定制的 AST 将具有相似语义的句法实体分组,并修剪掉具有少量或冗余语义的句法实体,优化了 CNN 定位模型的输入。

为进一步丰富代码的表示形式,文献[21]提出了一种基于知识图的缺陷定位方法 KGBugLocator,利用知识图嵌入对源代码实体的相互关系信息进行编码,然后分别通过缺陷特定的注意力和代码特定的注意力来丰富代码表示和缺陷报告表示。

为了更好地捕获源代码中具有附加语义的多维信息(如结构和功能性质),文献[19]提出了一种多实例学习框架的定位模型 CG-CNN,通过利用控制流程图(CFG)中的结构和顺序性质来增强定位模型输入的统一特征。

4.3.3 小结

本节从定位模型输入优化的角度梳理了现有的优化查询(缺陷报告)和优化语料库(待定位的源代码)两个方面的相关工作。总的来说,目前的研究工作对这两类输入的优化方法主要从两个角度着手。一种认为输入数据缺少对定位有用的信息(缺陷报告缺少相关术语,短长度源代码表示稀疏),于是采用扩展的方式对输入的表示进行重构。另一种认为输入数据中冗余信息过多(缺陷报告包含过多错误或无关信息,源代码语法结构冗余),于是采用降噪的方式对输入表示进行优化。从已有研究的进展来看,这种优化输入的策略已被广泛应用于两类定位模型。其中,使用深度学习模型的定位方法多利用源代码的语义特征(尤其是结构特征)来对语料库的表示进行优化,这可能是由于神经网络模型(尤其是 CNN)在学习源代码的结构语义特征(如 AST 的层次结构)上有相对较强的优势。

4.4 其他缺陷定位的方法

动态和静态的特征定位技术有互补作用[158]。针对不同类型缺陷,文献[70]提出同时考虑缺陷报告和程序谱来定位缺陷的技术 AML,自适应地创建了一个特定于缺陷的模型。部分研究[11,13]提出了两阶段的整体定位策略:首先使用分类模型判断缺陷报告是否可预测(或有效),如果可预测,则使用自动的定位工具进行定位,反之则直接采用人工的方式定位。

5　常用数据集

表3列出了面向软件缺陷报告的缺陷定位研究常用的公开可用的数据集的使用情况，包括数据集的名称及提供者、涉及项目数及其编程语言、缺陷报告数、定位级别和使用者。

从表3可以看出，面向软件缺陷报告的缺陷定位研究工作大多使用的是Zhou等[94](BugLocator)公开的数据集和Ye等[73](learning-to-rank)公开的数据集，其他公开数据集的可靠性还未得到足够的实验验证。几乎所有的数据集都是来自Java语言编写的项目，只有极少的研究使用了自建的其他编程语言的数据集。因此，现有的面向软件缺陷报告的缺陷定位研究的泛化性还未知，需要在更多不同来源和不同语言的数据集上进行实验验证。本领域现有研究的数据集的源代码单元以及定位粒度大多是在文件级别，源代码单元细粒度的(如方法级别)非常少。这是由于在软件项目中方法数量远多于文件数量，且代码方法不同于文件，不会在修复提交中直接给出，这使得检索代码方法更加困难。另外，和文件相比，方法的内容较少且区分度更低，导致方法级的缺陷定位普遍存在准确率低的问题。更多数据集信息也可参考前文综述网址。

表3　面向软件缺陷报告的定位方法常用数据集汇总

Table 3　Statistics of common data sets used in bug report-oriented bug localization

项目语言	定位级别	数据集	项目数	缺陷报告数	使用者
Java	File	[140]	1	390	[5,59,74,80,140,142]
		[131]	5	7401	[30,110,131,145]
		[94]	4	3479	[12,17,28-30,40,48,56,62,64-69,72,74-77,79,80,90,94,116,119,123]
		[73]	7	22747	[4,6-8,11,14,16,22,24,26-32,73,88,91,115,126]
		[120]	46	9459	[87,90,120,125,132]
		[31]	5	10754	[15,31]
		[9]	5	>300000	[9]
		[139]	7	7334	[121,122,139]
		[127]	5	1100	[127]
		[95]	2	395	[44,95]
		[144]	2	6716	[124,144]
	Class	[143]	5	114	[143]
	Method	[113]	1	1665	[113]
		[114]	7	1035	[114]
		[70]	4	157	[70,117]
		[137]	5	417	[41,61,78,137]
		[138]	17	825	[84,97]
C++	File	[144]	1	1368	[124,144]
C	File	[112]	5	7716	[112]
C#	File	[128]	20	878	[128,129]
Mixed	File	[130]	29	21253	[130]
		[96]	1	4650	[5,96]
		[136]	54	151161	[136]

6　评估指标

面向软件缺陷报告的缺陷定位方法中常用的评估指标如下：

(1) P(Precision Rate)，R(Recall Rate)，F_1(F_1-Measure)：对每个代码单元的定位也可看作二分类问题(是否为可疑代码单元)。P，R，F_1是最常用的分类任务评估指标。一个好的分类模型希望获得更高的P和R，并将其作为综合度量，F_1是P和R之间的权衡。

(2) Top N Rank：$TopN$指相关源代码单元排在结果列表的前N个的缺陷数量。给定一个缺陷报告，如果其中至少有一个需要修改以修复缺陷的源代码单元出现在排名列表的前N个位置，则该缺陷报告被视为已被定位到，并计入$TopN$结果中。$TopN$值越大，缺陷定位性能越好。Kochhar等[3]发现，95%以上的从业人员不会检查缺陷定位工具中超出前10个的结果。因此，N的值一般取10以内(如$N=1,5,10$)[62,94,112]。

(3) MRR(Mean Reciprocal Rank)：MRR是一种统计量度，取决于查询的排名列表中的首个相关文档[159]。缺陷定位中一个查询的倒数排名(RR)是首个被正确定位的源代码单元的排名的倒数，MRR是所有查询Q倒数排名的平均值。MRR值越大，缺陷定位性能越好。

(4) MAP(Mean Average Precision)：当一个查询可能有多个相关文档时，MAP为信息检索的质量提供一个单位数的度量。一组查询的MAP即所有查询的平均精度值的平均值。MAP值越大，缺陷定位性能越好。

(5) $Effort@k$，MAE(Mean Average Effort)，MFE(Mean First Effort)：从定位方法推荐的结果中找到真实的缺陷所在位置所需的工作量也是评价缺陷定位方法有效性的标准[115]。

$Effort@k$衡量查找缺陷报告的相关源代码单元所需的平均代码检查工作量。$Effort@k$代表在前k个建议下，为每个缺陷报告查找相关文件所需的平均代码检查工作量。在这里，k可以是1,2,3,依此类推。例如，Effort@3表示在前3个建议下查找每个缺陷报告的相关文件所需的平均代码检查工作。

MAE(MeanAverage Effort)源自标准度量平均精度均值(*MAP*),它定义为检查所有缺陷报告的所有相关文件所需要的平均工作量(*MAE*)。因此,与 *MAP* 相似,*MAE* 是检查每个缺陷报告的所有相关文件所需的平均工作量的平均值。

MFE(Mean First Effort)是为每个缺陷报告查找第一个相关文件所需的平均工作量。

(6)AUC(Area Under Curve),Cross Entropy Loss:这两项评估指标也是分类器的常用评估指标。

AUC 是接收器工作特性曲线(*ROC*)下的面积,该曲线图说明了阈值变化时二分类器系统的鉴别的诊断能力。交叉熵损失衡量(Cross Entropy Loss)的是分类器的性能,该分类器输出在[0,1]范围内。

(7)Time Efficiency:在基于深度学习的定位方法中,模型的计算时间也是一项重要评估指标。一般定义为一个缺陷报告定位出源代码单元的平均时间,其中包含模型的训练时间和预测时间[14,27,31]。

表 4 对以上评估指标的使用情况进行了汇总。Sangwan 等[33]曾指出,几乎所有的基于信息检索的模型都使用 *TopN*,*MRR*,*MAP* 这 3 个指标来进行定位效果评估。

表 4 面向软件缺陷报告的定位方法常用评估指标汇总

Table 4 Statistics of common evaluation metrics used in bug report-oriented bug localization

评估指标	使用者
P	[5,9,11-13,31,44,59-61,80,91,95-96,99,111,117-118,120,127]
R	[5,9,11,13,31,44,59-60,80-81,91,95-96,99,107,111,117-118,120,127]
*F*1	[9,11,31,60,111,118]
TopN	[6,8,10,12,14,17-30,32,37,46-49,51-57,62-70,72,74-77,79-80,83-91,93-95,98-101,104,108,112,114,117,119,121-124,126,128-129,131,133,135,144]
MRR	[4,6-8,14-15,17-19,21-27,30,32,40-41,46-48,51,53-54,56-57,61-63,65-69,72,74-77,79-81,84-88,90,93-94,97-101,103-108,112-113,117,119-123,125-127,132-133,135]
MAP	[4-8,14-22,24-32,40-41,43-44,46-48,50-52,56,59,62-63,65-70,72,74-82,84-88,90,94-98,100-108,110,112-113,117,119-123,125-130,132-133,135,142]
E	[37-38,41,43-44,47,49-50,71,83,92,99]
*Effort@*k	[115,124]
MAE	[115]
MFE	[115]
AUC	[28-29,126]
Cross Entropy Loss	[126]
Time Efficiency	[14,27,31,116]

从表中也可以看到,大多数的面向软件缺陷报告的缺陷定位方法都使用了这 3 种评估指标。从 *MRR* 和 *MAP* 指标的定义可以看出,这两种常用指标可以评估定位方法的整体平均性能,能够相对公平地评价各类定位方法。但以上 3 种主流的评估指标未考虑推荐代码单元的大小对实际缺陷定位效率的影响。实际上,如果建议的是行数较多的代码单元,则开发人员通常将需要更多的代码检查工作来定位缺陷。为了评估所提出的定位方法的效率,文献[14,27,31]计算了定位方法的必要计算时间,即模型的训练时长和预测时长。评估结果表明,被测方法都能达到实用的时间效率。因此,除了整体评估指标,对定位方法的实际效率的评估也具有重要的参考价值。

7 挑战与展望

7.1 定位模型

(1)如何有效结合多种定位模型。词汇差异问题会降低传统的基于 IR 的定位的准确率。而深度学习使用文本语义相似性有助于做出面向上下文的决策。从总体上看(见图 1),目前仅提出了较少的基于深度学习的软件缺陷定位模型。在该领域中还有很多方面未探索到,例如研究解决词嵌入难以学习缺陷报告或代码段的句子语义的问题、解决因超长代码单元引起的 CNN 维度爆炸的问题、如何学习代码元素在程序结构上的长依赖关系等。另外,直觉上,结合词汇相似度和语义相似度能够提升定位模型的准确率,但 4.1.4 节分析了现有的组合模型并未明显提升定位模型的性能,因此如何有效地结合两类模型来充分提高定位方法的性能是本领域面临的一个重要挑战。

(2)提高定位模型的准确率。学习模型的准确性很大程度上受到训练数据规模的影响。Lee 等[120]指出,对缺陷定位方法的性能评估通常集中在规模有限的旧项目集上。由于主体选择的偏差,定位模型可能存在过拟合问题[75]。因此,训练数据不足会导致定位方法的准确性降低。而现有的相关系统中存在着大量的无监督的数据(如已修复历史缺陷的报告和源代码、API 文档、说明文档等)。因此,除了充分有效地利用软件存储库中相关的其他隐藏信息之外,还可考虑从领域数据中挖掘可用信息。语言模型先预训练出词汇的向量表示,可以弥补下游定位任务训练数据不足的问题。因此可以考虑使用预训练模型,利用这些大规模的无监督的数据来帮助提高基于深度学习的定位方法的准确性。

(3)解决定位模型泛化性不足的问题。现有缺陷定位模型大多是在相同的软件项目内训练和验证的。另外,实验数据大多使用个别公开的数据集[73,94],且数据集大多来自 Java 语言编写的软件项目。因此,现有缺陷定位技术在跨项目和跨语言任务上的有效性和适用性仍有待进一步的实证研究。而不同编程语言编写的源代码文件在更抽象的层次上具有相同的特征,例如代码结构和代码单元的概念类型(如 Java 的方法和 C 的函数)。已有研究[30]表明,迁移学习的方法可以学习源和目标之间共享的潜在特征表示,从而缓解冷启动问题。因此,可考虑利用此类方法来学习不同语言编写的源代码单元共同的、潜在的特征,以提高定位模型的泛化性。

7.2 模型输出优化

如何有效结合多种来源的数据特征。已有研究分析表明,除了缺陷报告和源代码文件的文本,附加使用其他来源的数据特征有助于提高缺陷定位的准确性[117,121-123]。现有研究对定位输出的优化大多是先对不同的数据源分别计算得分再进行简单的组合,这可能会影响查询和语料之间相关度的计算。已有经验研究[123]表明,探索更合适的计算算法来有效组合多个相关度得分可以提高缺陷定位的准确率。此外,少数研究[22,24]利用神经网络自动集成不同维度的相关度,并

迭代地学习不同维度特征和缺陷位置之间的非线性关系，从而辅助提高了定位准确率。因此，未来可以研究更有效的来自多源特征的相似度组合计算算法，或者探索合适的模型或技术来处理不同来源的数据特征，并转换成统一的表示以计算最终相关度得分，从而有效提高定位方法的准确性。

7.3　模型输入优化

(1)重构输入的方法有待深入研究。输入的训练数据的质量会影响定位模型的准确性。已有研究[97,102,108]表明，输入缺陷报告的噪音过多或者有用信息不足会降低缺陷定位模型的准确率。通过查询重构、有效语料库构建的方式可以优化输入，从而提高定位的准确性[96-109]。目前已有少数研究[96-109]使用查询重构技术来构建有效的查询或语料库，并给出了查询重构的策略，但仅仅在有限的几个缺陷定位技术上进行了验证，因此查询重构策略的构建和实证分析仍有待进一步的研究。

(2)输入的特征表示有待优化。不同语言的数据通常具有不同的表示形式，例如源代码单元之间通过调用或实现而存在相关关系，利用这些特征有助于提升定位模型的准确率[8,100]。因此，未来可以考虑在输入向量的表示中增加其他特征(如文件或方法间的调用依赖关系)来增强语义表达能力。特别地，当源代码中蕴含着丰富的非线性结构信息(如AST结构、控制流、数据流等)，可考虑在未来的定位研究中使用图形的方式来表示源代码输入，并利用合适的学习模型抽取这些语义特征，进而提高定位的准确性。

7.4　数据集

数据集的质量会影响定位性能。软件演化过程中，许多代码单元会随着项目版本变化而发生修改、新增、删除，而传统文件级别的定位方法往往针对项目的一个“初始版本”进行定位[73]，许多新修改的代码单元无法被定位到，可能会造成较大误差。因此，数据集的质量会在很大程度上影响定位模型的性能[110-111,118,131,145]。如何过滤无效缺陷报告和选择准确版本的代码以获取高质量的数据集也是本研究领域面临的一项挑战和未来值得探索的方向。另一方面，现有缺陷定位方法大多是在旧的经典数据集上使用各自的方式来对数据集进行处理，因而数据集的不同导致了不同的定位性能。此外，针对深度学习模型，超大规模的数据集会增加模型训练所需的成本[126]。为了对定位技术实际性能进行的全面分析，可以考虑使用统一的方法处理输入数据，针对大规模数据还应考虑使用降维的技术，构建合适规模的数据集。

7.5　评估指标

定位实验评估标准具有局限性。文献[160]根据对开发人员的调查发现，如果调试工具无法帮助输出排名列表的前几个确定缺陷的根本原因，那么开发人员会认为该调试工具无用。而几乎所有的现有定位方法使用的评估标准都是定位方法在一组缺陷报告测试集上的平均性能值(*MAP* 和 *MRR*)。可见这种整体评价指标尚不能满足实际的工业需求。实际上，如果建议的是较大的文件，那么开发人员通常需要更多的代码检查工作来定位缺陷[115]。因此，应考虑更全面的定位评估指标。例如，评估定位方法的工作量有助于开发人员判断定位方法的实际有效性[14,27,31]。

7.6　其他方面

(1)优化整体定位策略。实践中，开发人员很难决定哪种工具对给定的缺陷报告有效[118]。甚至对于某些缺陷报告，没有任何缺陷定位工具是有用的。即使是最新的缺陷定位工具，也会输出许多排名列表，其中缺陷文件在列表中的位置可能非常低。因此，先预测的方式是整个缺陷修复任务值得探索的方向。首先预测定位工具的有效性，再采用定位方法定位缺陷。

(2)更细粒度的定位。现有的定位粒度大多是在文件级别，对于行数较多的源代码文件，修复者使用文件级定位工具，之后还需人工阅读代码文件以查找更细粒度的缺陷位置。一项实证研究[126]表明，半数以上的软件从业者更偏爱方法级别的定位工具。近年来，已有少数研究提出更细粒度的定位缺陷[17,75,100]，并取得了可观的实验效果。对于更改级别的缺陷定位，开发人员不仅能定位缺陷，而且还能方便地识别导致将缺陷引入系统的代码更改[17]。因此，考虑细粒度的缺陷定位，不仅能够提升定位的效率，节省人工成本，还能为后续缺陷修复任务提供帮助。

结束语　面向缺陷报告的软件缺陷定位方法数据源丰富，易获取，人工成本低，且对不同语言的适用性相对较好，近年来已成为软件工程领域研究的热点。本文从定位模型的研究、对模型输出的优化研究和对模型输入的优化研究3个层面对近十年来面向软件缺陷报告的缺陷定位的研究工作进行了梳理和总结，其中，定位模型方面重点介绍和分析了基于深度学习的定位模型。本文还对该领域的常用数据集和评估指标进行了简要汇总和分析。根据目前的研究进展，本文从上述3个层面以及数据集和评估指标方面探讨了该领域面临的挑战和未来可能的研究趋势。

参 考 文 献

[1] BRAUDE E J, BERNSTEIN M E. Software engineering: modern approaches[M]. Long Grove: Waveland Press, 2016.

[2] ABREU R, ZOETEWEIJ P, GOLSTEIJN R, et al. A practical evaluation of spectrum-based fault localization[J]. Journal of Systems and Software, 2009, 82(11): 1780-1792.

[3] KOCHHAR P S, XIA X, LO D, et al. Practitioners' expectations on automated fault localization[C]// Proceedings of the 25th International Symposium on Software Testing and Analysis. New York: ACM, 2016: 165-176.

[4] YE X, SHEN H, MA X, et al. From word embeddings to document similarities for improved information retrieval in software engineering[C]// Proceedings of the 38th International Conference on Software Engineering. New York: ACM, 2016: 404-415.

[5] AKBAR S, KAK A. SCOR: source code retrieval with semantics and order[C]// 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). Piscataway, NJ: IEEE, 2019: 1-12.

[6] XIAO Y, KEUNG J, BENNIN K E, et al. Machine translation-based bug localization technique for bridging lexical gap[J]. Information and Software Technology, 2018, 99: 58-61.

[7] XIAO Y, KEUNG J. Improving bug localization with character-level convolutional neural network and recurrent neural network [C]// 2018 25th Asia-Pacific Software Engineering Conference (APSEC). 2018: 703-704.

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

[9] DISTANTE D, FARALLI S. A two-phase bug localization approach based on multi-layer perceptrons and distributional features[C]// International Conference on Computational Science and Its Applications. Berlin: Springer, 2019: 518-532.

[10] GUPTA R, KANADE A, SHEVADE S. Deep learning for bug-localization in student programs[J]. arXiv: 1905.12454, 2019.

[11] YE X, FANG F, WU J, et al. Bug Report Classification using LSTM architecture for more accurate software defect locating [C]// 2018 17th IEEE International Conference on Machine Learning and Applications(ICMLA). Piscataway, NJ: IEEE, 2018: 1438-1445.

[12] LIU G, LU Y, SHI K, et al. Mapping bug reports to relevant source code files based on the vector space model and word embedding[J]. IEEE Access, 2019, 7: 78870-78881.

[13] KIM D, TAO Y, KIM S, et al. Where should we fix this bug? a two-phase recommendation model[J]. IEEE Transactions on Software Engineering, 2013, 39(11): 1597-1610.

[14] UNENO Y, MIZUNO O, CHOI E H. Using a distributed representation of words in localizing relevant files for bug reports [C]// 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS). Piscataway, NJ: IEEE, 2016: 183-190.

[15] XIAO Y, KEUNG J, MI Q, et al. Bug localization with semantic and structural features using convolutional neural network and cascade forest[C]// Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018. ACM, 2018: 101-111.

[16] BARBOSA J R, MARCACINI R M, BRITTO R, et al. BULNER: Bug Localization with word embeddings and network regularization[J]. arXiv: 1908.09876, 2019.

[17] LOYOLA P, GAJANANAN K, SATOH F. Bug localization by learning to rank and represent bug inducing changes[C]// Proceedings of the 27th ACM International Conference on Information and Knowledge Management. New York: ACM, 2018: 657-665.

[18] JIANG B, LIU P, XU J. A Deep Learning Approach to Locate Buggy Files[C]// 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies(DESSERT). IEEE, 2020: 219-223.

[19] HUO X, LI M, ZHOUZ H. Control flow graph embedding based on multi-instance decomposition for bug localization[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2020: 4223-4230.

[20] ZHU Z, LI Y, TONG H, et al. CooBa: Cross-project Bug Localization via Adversarial Transfer Learning[C]// IJCAI. 2020: 3565-3571.

[21] ZHANG J, XIE R, YE W, et al. Exploiting code knowledge graph for bug localization via bi-directional attention[C]// Proceedings of the 28th International Conference on Program Comprehension. 2020: 219-229.

[22] WANG B, XU L, YAN M, et al. Multi-dimension convolutional neural network for bug localization[J]. IEEE Transactions on Services Computing, 2020, 15(3): 1649-1663.

[23] PRADEL M, MURALI V, QIAN R, et al. Scaffle: bug localization on millions of files[C]// Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM, 2020: 225-236.

[24] CHENG S, YAN X, KHAN A A. A Similarity Integration Method based Information Retrieval and Word Embedding in Bug Localization[C]// 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS). Piscataway, NJ: IEEE, 2020: 180-187.

[25] SANGLE S, MUVVA S, CHIMALAKONDA S, et al. DRAST--A Deep Learning and AST Based Approach for Bug Localization [J]. arXiv: 2011.03449, 2020.

[26] LAM A N, NGUYEN A T, NGUYEN H A, et al. Combining deep learning with information retrieval to localize buggy files for bug reports (n)[C]// 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). Piscataway, NJ: IEEE, 2015: 476-481.

[27] LAM A N, NGUYEN A T, NGUYEN H A, et al. Bug localization with combination of deep learning and information retrieval [C]// 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC). Piscataway, NJ: IEEE, 2017: 218-229.

[28] HUO X, LI M, ZHOU Z H. Learning unified features from natural and programming languages for locating buggy source code [C]// IJCAI. 2016: 1606-1612.

[29] HUO X, LI M. Enhancing the Unified Features to Locate Buggy Files by Exploiting the Sequential Nature of Source Code[C]// IJCAI. 2017: 1909-1915.

[30] HUO X, THUNG F, LI M, et al. Deep transfer bug localization [J]. IEEE Transactions on Software Engineering, 2019, 47(7): 1368-1380.

[31] XIAO Y, KEUNG J, MI Q, et al. Improving bug localization with an enhanced convolutional neural network[C]// 2017 24th Asia-Pacific Software Engineering Conference (APSEC). Piscataway, NJ: IEEE, 2017: 338-347.

[32] 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.

[33] SANGWAN O P. Review of text mining techniques for software bug localization[C]// 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 2019: 208-211.

[34] ZHANG Y, LIU J K, XIA X, et al. Information retrieval-based based bug localization: a literature review [J]. Journal of Software, 2020, 31(8): 2432-2452.

[35] GUO Z Q, ZHOU H C, LIU S R, et al. Information retrieval based bug localization: research problem, progress, and challen-

ges[J]. Journal of Software，2020，31(9)：2826-2854.

[36] LI Z L，CHEN X，JIANG Z W，et al. Survey of information retrieval-based software bug localization. [J]. Journal of Software，2021，32(2)：247-276.

[37] LUKINS S K，KRAFT N A，ETZKORN L H. Source code retrieval for bug localization using latent dirichlet allocation[C]// 2008 15th Working Conference on Reverse Engineering. Piscataway，NJ：IEEE，2008：155-164.

[38] NICHOLS B D. Augmented bug localization using past bug information[C]// Proceedings of the 48th Annual Southeast Regional Conference. New York：ACM，2010：1-6.

[39] DE ALMEIDA MAIA M，COSTA A，DA SILVA I R. On the Influence of Latent Semantic Analysis Parameterization for Bug Localization[J]. Revista de Informática Teórica e Aplicada，2013，20(3)：49-76.

[40] WANG S，LO D，LAWALL J. Compositional vector space models for improved bug localization[C]// 2014 IEEE International Conference on Software Maintenance and Evolution. Piscataway，NJ：IEEE，2014：171-180.

[41] MORENO L，TREADWAY J J，MARCUS A，et al. On the use of stack traces to improve text retrieval-based bug localization [C]// 2014 IEEE International Conference on Software Maintenance and Evolution. IEEE，2014：151-160.

[42] PATHAK D P，DHARAVATH S. Automation framework for bug localization using information retrieval techniques[J]. International Journal of Innovative Research in Computer and Communication Engineering，2015，3(6)：5720-5727.

[43] KUMAR D，SHARMA R. Bug localization using ldacg approach [J]. International Journal of Engineering Research and Technology，2015，4(5)：155-160.

[44] RAO S，MEDEIROS H，KAK A. Comparing incremental latent semantic analysis algorithms for efficient retrieval from software libraries for bug localization[J]. ACM SIGSOFT Software Engineering Notes，2015，40(1)：1-8.

[45] RAHMAN S，SAKIB K. An Appropriate Method Ranking Approach for Localizing Bugs using Minimized Search Space [C]// ENASE. 2016：303-309.

[46] RAHMAN S，RAHMAN M M，SAKIB K. An improved method level bug localization approach using minimized code space[C]// International Conference on Evaluation of Novel Approaches to Software Engineering. Berlin：Springer，2016：179-200.

[47] GORE A，CHOUBEY S D，GANGRADE K. Improved Bug Localization Technique Using Hybrid Information Retrieval Model [C]// International Conference on Distributed Computing and Internet Technology. Berlin：Springer，2016：127-131.

[48] KILINÇ D，YÜCALAR F，BORANDAĞ E，et al. Multi-level reranking approach for bug localization[J]. Expert Systems，2016，33(3)：286-294.

[49] LUKINS S K，KRAFT N A，ETZKORN L H. Bug localization using latent dirichlet allocation[J]. Information and Software Technology，2010，52(9)：972-990.

[50] SHARMA T，SHARMA K，SHARMA T. Software bug localization using pachinko allocation model[C]// 2016 3rd International Conference on Computing for Sustainable Global Development(INDIACom). IEEE，2016：3603-3608.

[51] RAHMAN S，RAHMAN M M，SAKIB K. A Statement Level Bug Localization Technique using Statement Dependency Graph [C]// ENASE. 2017：171-178.

[52] ZHANG X，YAO Y，WANG Y，et al. Exploring metadata in bug reports for bug localization[C]// 2017 24th Asia-Pacific Software Engineering Conference(APSEC). Piscataway，NJ：IEEE，2017：328-337.

[53] WANG Y，YAO Y，TONG H，et al. Enhancing supervised bug localization with metadata and stack-trace[J]. Knowledge and Information Systems，2020，62(6)：2461-2484.

[54] WANG Y，YAO Y，TONG H，et al. Bug localization via supervised topic modeling[C]// 2018 IEEE International Conference on Data Mining(ICDM). Piscataway，NJ：IEEE，2018：607-616.

[55] NGUYEN A T，NGUYEN T T，AL-KOFAHI J，et al. A topic-based approach for narrowing the search space of buggy files from a bug report[C]// 2011 26th IEEE/ACM International Conference on Automated Software Engineering(ASE 2011). Piscataway，NJ：IEEE，2011：263-272.

[56] SWE K E E，OO H M. Source code retrieval for bug localization using bug report[C]// 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing(ICCP). IEEE，2019：241-247.

[57] KHATIWADA S，TUSHEV M，MAHMOUD A. On Combining IR Methods to Improve Bug Localization[C]// Proceedings of the 28th International Conference on Program Comprehension. New York：ACM，2020：252-262.

[58] BEARD M. Extending bug localization using information retrieval and code clone location techniques[C]// 2011 18th Working Conference on Reverse Engineering. Piscataway，NJ：IEEE，2011：425-428.

[59] SISMAN B，KAK A C. Incorporating version histories in information retrieval based bug localization[C]// 2012 9th IEEE Working Conference on Mining Software Repositories(MSR). Piscataway，NJ：IEEE，2012：50-59.

[60] MOIN A H，KHANSARI M. Bug localization using revision log analysis and open bug repository text categorization[C]// IFIP International Conference on Open Source Systems. Berlin：Springer，2010：188-199.

[61] DAVIES S，ROPER M，WOOD M. Using bug report similarity to enhance bug localisation[C]// 2012 19th Working Conference on Reverse Engineering. Piscataway，NJ：IEEE，2012：125-134.

[62] SAHA R K，LEASE M，KHURSHID S，et al. Improving bug localization using structured information retrieval[C]// 2013 28th IEEE/ACM International Conference on Automated Software Engineering(ASE). Piscataway，NJ：IEEE，2013：345-355.

[63] TANTITHAMTHAVORN C，TEEKAVANICH R，IHARA A，et al. Mining A change history to quickly identify bug locations：A case study of the Eclipse project[C]// 2013 IEEE International Symposium on Software Reliability Engineering Workshops(ISSREW). Piscataway，NJ：IEEE，2013：108-113.

[64] TANTITHAMTHAVORN C，IHARA A，MATSUMOTO K. Using co-change histories to improve bug localization performance[C]// 2013 14th ACIS International Conference on Soft-

ware Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. Piscataway, NJ: IEEE, 2013: 543-548.

[65] WONG C P, XIONG Y, ZHANG H, et al. Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis[C]// 2014 IEEE International Conferenceon Software Maintenance and Evolution. Piscataway, NJ: IEEE, 2014: 181-190.

[66] WANG S, LO D. Version history, similar report, and structure: Putting them together for improved bug localization[C]// Proceedings of the 22nd International Conference on Program Comprehension. 2014: 53-63.

[67] RAHMAN S, GANGULY K K, SAKIB K. An improved bug localization using structured information retrieval and version history[C]// International Conference on Computer and Information Technology. 2016: 190-195.

[68] YOUM K C, AHN J, KIM J, et al. Bug localization based on code change histories and bug reports[C]// 2015 Asia-Pacific Software Engineering Conference(APSEC). Piscataway, NJ: IEEE, 2015: 190-197.

[69] YOUM K C, AHN J, LEE E. Improved bug localization based on code change histories and bug reports[J]. Information and Software Technology, 2017, 82: 177-192.

[70] LE T D B, OENTARYO R J, LO D. Information retrieval and spectrum based bug localization: Better together[C]// Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. ACM, 2015: 579-590.

[71] SHAO P. Combining information retrieval modules and structural information for source code bug localization and feature location[M]. The University of Alabama, Ann Arbor: ProQuest, 2011.

[72] TONG Y, ZHOU Y, FANG L, et al. Towards a novel approach for defect localization based on part-of-speech and invocation [C]// Proceedings of the 7th Asia-Pacific Symposium on Internetware. ACM, 2015: 52-61.

[73] 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. New York: ACM, 2014: 689-699.

[74] WANG S, LO D. Amalgam+: Composing rich information sources for accurate bug localization[J]. Journal of Software: Evolution and Process, 2016, 28(10): 921-942.

[75] WEN M, WU R, CHEUNG S C. Locus: Locating bugs from software changes[C]// 2016 31st IEEE/ACM International Conference on Automated Software Engineering(ASE). Piscataway, NJ: IEEE, 2016: 262-273.

[76] ZHOU Y, TONG Y, CHEN T, et al. Augmenting bug localization with part-of-speech and invocation[J]. International Journal of Software Engineering and Knowledge Engineering, 2017, 27(6): 925-949.

[77] GHARIBI R, RASEKH A H, SADREDDINI M H. Locating relevant source files for bug reports using textual analysis[C]// 2017 International Symposium on Computer Science and Software Engineering Conference (CSSE). Piscataway, NJ: IEEE, 2017: 67-72.

[78] TAKAHASHI A, SAE-LIM N, HAYASHI S, et al. A preliminary study on using code smells to improve bug localization [C]// Proceedings of the 26th Conference on Program Comprehension. New York: ACM, 2018: 324-327.

[79] SWE K E E, OO H M. Bug Localization Approach Using Source Code Structure with Different Structure Fields[C]// 2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications(SERA). Piscataway, NJ: IEEE, 2018: 159-164.

[80] KHATIWADA S, TUSHEV M, MAHMOUD A. Just enough semantics: An information theoretic approach for IR-based software bug localization[J]. Information and Software Technology, 2018, 93: 45-57.

[81] ZHANG T, HU W, LUO X, et al. A commit messages-based bug localization for android applications[J]. International Journal of Software Engineering and Knowledge Engineering, 2019, 29(4): 457-487.

[82] LAL S, SUREKA A. A static technique for fault localization using character n-gram based information retrieval model[C]// Proceedings of the 5th India Software Engineering Conference. New York: ACM, 2012: 109-118.

[83] SUN X, ZHOU W, LI B, et al. Bug localization for version issues with defect patterns[J]. IEEE Access, 2019, 7: 18811-18820.

[84] HOANG T, OENTARYO R J, LE T D B, et al. Network-clustered multi-modal bug localization[J]. IEEE Transactions on Software Engineering, 2018, 45(10): 1002-1023.

[85] RATH M, LO D, MÄDER P. Analyzing requirements and traceability information to improve bug localization[C]// Proceedings of the 15th International Conference on Mining Software Repositories. 2018: 442-453.

[86] DILSHENER T, WERMELINGER M, YU Y. Locating bugs without looking back[J]. Automated Software Engineering, 2018, 25(3): 383-434.

[87] KOYUNCU A, BISSYANDÉ T F, KIM D, et al. D&c: A divide-and-conquer approach to ir-based bug localization[J]. arXiv: 1902.02703, 2019.

[88] YE X, BUNESCU R, LIU C. Mapping bug reports to relevant files: A ranking model, a fine-grained benchmark, and feature evaluation[J]. IEEE Transactions on Software Engineering, 2015, 42(4): 379-402.

[89] ASATO M, AMAN H, AMASAKI S, et al. A Mahalanobis Distance-Based Integration of Suspicious Scores For Bug Localization[C]// 2020 27th Asia-Pacific Software Engineering Conference(APSEC). 2020: 475-479.

[90] LI Z, JIANG Z, CHEN X, et al. Laprob: A Label propagation-Based software bug localization method[J]. Information and Software Technology, 2021, 130: 106410.

[91] ALMHANA R, KESSENTINI M, MKAOUER W. Method-level bug localization using hybrid multi-objective search[J]. Information and Software Technology, 2021, 131: 106474.

[92] SHAO P, ATKISON T, KRAFT N A, et al. Combining lexical and structural information for static bug localisation[J]. International Journal of Computer Applications in Technology, 2012, 44(1): 61-71.

[93] BANGCHAROENSAP P，IHARA A，KAMEI Y，et al. Locating source code to be fixed based on initial bug reports-a case study on the eclipse project[C]// 2012 Fourth International Workshop on Empirical Software Engineering in Practice. Piscataway，NJ：IEEE，2012：10-15.

[94] ZHOU J，ZHANG H，LO D. Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports[C]// 2012 34th International Conference on Software Engineering (ICSE). Piscataway，NJ：IEEE，2012：14-24.

[95] RAO S，MEDEIROS H，KAK A. An incremental update framework for efficient retrieval from software libraries for bug localization[C]// 2013 20th Working Conference on Reverse Engineering(WCRE). IEEE，2013：62-71.

[96] SISMAN B，KAK A C. Assisting code search with automatic query reformulation for bug localization[C]// 2013 10th Working Conference on Mining Software Repositories (MSR). IEEE，2013：309-318.

[97] CHAPARRO O，FLOREZ J M，MARCUS A. Using observed behavior to reformulate queries during text retrieval-based bug localization[C]// 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). Piscataway，NJ：IEEE，2017：376-387.

[98] KIM M，LEE E. ManQ：Many-objective optimization-based automatic query reduction for IR-based bug localization[J]. Information and Software Technology，2020，125：106334.

[99] MILLS C，PARRA E，PANTIUCHINA J，et al. On the relationship between bug reports and queries for text retrieval-based bug localization[J]. Empirical Software Engineering，2020，25(5)：3086-3127.

[100] ZHANG W，LI Z，WANG Q，et al. FineLocator：A novel approach to method-level fine-grained bug localization by query expansion[J]. Information and Software Technology，2019，110：121-135.

[101] ZHANG W，LI Z Q，DU Y H，et al. Fine-grained software bug location approach at method level[J]. Journal of Software，2019，30(2)：195-210.

[102] RAHMAN M M，ROY C. Poster：improving bug localization with report quality dynamics and query reformulation[C]// 2018 IEEE/ACM 40th International Conference on Software Engineering：Companion (ICSE-Companion). Piscataway，NJ：IEEE，2018：348-349.

[103] LAWRIE D，BINKLEY D. On the value of bug reports for retrieval-based bug localization[C]// 2018 IEEE International Conference on Software Maintenance and Evolution(ICSME). IEEE，2018：524-528.

[104] RAHMAN M M，ROY C K. Improving ir-based bug localization with context-aware query reformulation[C]// Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2018：621-632.

[105] MILLS C，PANTIUCHINA J，PARRA E，et al. Are bug reports enough for text retrieval-based bug localization? [C]// 2018 IEEE International Conference on Software Maintenance and Evolution(ICSME). IEEE，2018：381-392.

[106] CHAPARRO O，FLOREZ J M，MARCUS A. Using bug descriptions to reformulate queries during text-retrieval-based bug localization[J]. Empirical Software Engineering，2019，24(5)：2947-3007.

[107] RAHMAN M M. Supporting code search with context-aware，analytics-driven，effective query reformulation[C]// 2019 IEEE/ACM 41st International Conference on Software Engineering：Companion Proceedings(ICSE-Companion). Piscataway，NJ：IEEE，2019：226-229.

[108] KIM M，LEE E. A novel approach to automatic query reformulation for ir-based bug localization[C]// Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. New York：ACM，2019：1752-1759.

[109] CHAPARRO O. Improving bug reporting，duplicate detection，and localization[C]// 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). IEEE，2017：421-424.

[110] KOCHHAR P S，TIAN Y，LO D. Potential biases in bug localization：Do they matter? [C]// Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering. New York：ACM，2014：803-814.

[111] LE T D B，THUNG F，LO D. Predicting effectiveness of ir-based bug localization techniques[C]// 2014 IEEE 25th International Symposium on Software Reliability Engineering. Piscataway，NJ：IEEE，2014：335-345.

[112] SAHA R K，LAWALL J，KHURSHID S，et al. On the effectiveness of information retrieval based bug localization for c programs[C]// 2014 IEEE International Conference on Software Maintenance and Evolution. IEEE，2014：161-170.

[113] ALDUAILIJ M，AL-DUAILEJ M. Performance evaluation of information retrieval models in bug localization on the method level[C]// 2015 International Conference on Collaboration Technologies and Systems(CTS). Piscataway，NJ：IEEE，2015：305-313.

[114] MORENO L，BAVOTA G，HAIDUC S，et al. Query-based configuration of text retrieval solutions for software engineering tasks[C]// Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. New York：ACM，2015：567-578.

[115] ZHAO F，TANG Y，YANG Y，et al. Is learning-to-rank cost-effective in recommending relevant files for bug localization? [C]// 2015 IEEE International Conference on Software Quality，Reliability and Security. Piscataway，NJ：IEEE，2015：298-303.

[116] SHI Z，KEUNG J，BENNIN K E，et al. A Strategy to Determine When to Stop Using Automatic Bug Localization[C]// 2016 IEEE 40th Annual Computer Software and Applications Conference(COMPSAC). 2016，1：185-190.

[117] DAO T，ZHANG L，MENG N. How does execution information help with information-retrieval based bug localization? [C]// 2017 IEEE/ACM 25th International Conference on Program Comprehension(ICPC). Piscataway，NJ：IEEE，2017：241-250.

[118] LE T D B，THUNG F，LO D. Will this localization tool be effective for this bug? Mitigating the impact of unreliability of infor-

mation retrieval based bug localization tools[J]. Empirical Software Engineering, 2017, 22(4): 2237-2279.

[119]KIM M, LEE E. Poster: Are Information Retrieval-Based Bug Localization Techniques Trustworthy? [C]// 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion). Piscataway, NJ: IEEE, 2018: 248-249.

[120]LEE J, KIM D, BISSYANDÉ T F, et al. Bench4bl: reproducibility study on the performance of ir-based bug localization[C]// Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York: ACM, 2018: 61-72.

[121]RATH M, MÄDER P. Influence of structured information in bug report descriptions on ir-based bug localization[C]// 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Piscataway, NJ: IEEE, 2018: 26-32.

[122]RATH M, MÄDER P. Structured information in bug report descriptions—influence on IR-based bug localization and developers[J]. Software Quality Journal, 2019, 27(3): 1315-1337.

[123]SHI Z, KEUNG J, BENNIN K E, et al. Comparing learning to rank techniques in hybrid bug localization[J]. Applied Soft Computing, 2018, 62: 636-648.

[124]TANTITHAMTHAVORN C, ABEBE S L, HASSAN A E, et al. The impact of IR-based classifier configuration on the performance and the effort of method-level bug localization[J]. Information and Software Technology, 2018, 102: 160-174.

[125]AMASAKI S, AMAN H, YOKOGAWA T. A Comparative Study of Vectorization Methods on BugLocator[C]// 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2019: 236-243.

[126]POLISETTY S, MIRANSKYY A, BAŞAR A. On usefulness of the deep-learning-based bug localization models to practitioners [C]// Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering. New York: ACM, 2019: 16-25.

[127]RAHMAN M M, CHAKRABORTYS, KAISER G, et al. Toward Optimal Selection of Information Retrieval Models for Software Engineering Tasks [C] // 2019 19th International Working Conference on Source Code Analysis and Manipulation (SCAM). Piscataway, NJ: IEEE, 2019: 127-138.

[128]GARNIER M, GARCIA A. On the evaluation of structured information retrieval-based bug localization on 20 C # projects [C]// Proceedings of the 30th Brazilian Symposium on Software Engineering. ACM, 2016: 123-132.

[129]GARNIER M, FERREIRA I, GARCIA A. On the influence of program constructs on bug localization effectiveness[J]. Journal of Software Engineering Research and Development, 2017, 5(1): 1-29.

[130]AKBAR S A, KAK A C. A large-scale comparative evaluation of IR-based tools for bug localization[C]// Proceedings of the 17th International Conference on Mining Software Repositories. ACM, 2020: 21-31.

[131]HERZIG K, JUST S, ZELLER A. It's not a bug, it's a feature: how misclassification impacts bug prediction[C]// 2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013: 392-401.

[132]AMASAKI S, AMAN H, YOKOGAWA T. On the Effects of File-level Information on Method-level Bug Localization[C]// 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Piscataway, NJ: IEEE, 2020: 314-321.

[133]XIA X, LO D, WANG X, et al. Cross-language bug localization [C]// Proceedings of the 22nd International Conference on Program Comprehension. New York: ACM, 2014: 275-278.

[134]THUNG F, LE T D B, KOCHHAR P S, et al. Buglocalizer: Integrated tool support for bug localization[C]// Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York: ACM, 2014: 767-770.

[135]WANG X, LO D, XIA X, et al. BOAT: an experimental platform for researchers to comparatively and reproducibly evaluate bug localization techniques[C]// Companion Proceedings of the 36th International Conference on Software Engineering. New York: ACM, 2014: 572-575.

[136]MUVVA S, RAO A E, CHIMALAKONDA S. BuGL--A Cross-Language Dataset for Bug Localization[J]. arXiv: 2004. 08846, 2020.

[137]DIT B, REVELLE M, GETHERS M, et al. Feature location in source code: a taxonomy and survey[J]. Journal of Software: Evolution and Process, 2013, 25(1): 53-95.

[138]JUST R, JALALI D, ERNST M D. Defects4J: A database of existing faults to enable controlled testing studies for Java programs[C]// Proceedings of the 2014 International Symposium on Software Testing and Analysis. New York: ACM, 2014: 437-440.

[139]RATH M, REMPEL P, MÄDER P. The ilmseven dataset[C]// 2017 IEEE 25th International Requirements Engineering Conference (RE). 2017: 516-519.

[140]DALLMEIER V, ZIMMERMANN T. Extraction of bug localization benchmarks from history[C]// Proceedings of the twenty-second IEEE/ACM International Conference on Automated Software Engineering. New York: ACM, 2007: 433-436.

[141]BEARD M, KRAFT N, ETZKORN L, et al. Measuring the accuracy of information retrieval based bug localization techniques [C]// 2011 18th Working Conference on Reverse Engineering. Piscataway, NJ: IEEE, 2011: 124-128.

[142]RAO S, KAK A. Retrieval from software libraries for bug localization: a comparative study of generic and composite text models [C]// Proceedings of the 8th Working Conference on Mining Software Repositories. New York: ACM, 2011: 43-52.

[143]MORENO L, BANDARA W, HAIDUC S, et al. On the relationship between the vocabulary of bug reports and source code [C]// 2013 IEEE International Conference on Software Maintenance. IEEE, 2013: 452-455.

[144]THOMAS S W, NAGAPPAN M, BLOSTEIN D, et al. The impact of classifier configuration and classifier combination on bug localization[J]. IEEE Transactions on Software Engineering, 2013, 39(10): 1427-1443.

[145]KOCHHAR P S，LE T D B，LO D. It's not a bug，it's a feature：does misclassification affect bug localization? [C]// Proceedings of the 11th Working Conference on Mining Software Repositories. 2014：296-299.

[146]POSHYVANYK D，MARCUS A，RAJLICH V，et al. Combining probabilistic ranking and latent semantic indexing for feature identification[C]// 14th IEEE International Conference on Program Comprehension (ICPC'06). Piscataway，NJ：IEEE，2006：137-148.

[147]LEVY O，GOLDBERG Y. Neural word embedding as implicit matrix factorization[J]. Advances in Neural Information Processing Systems，2014，27：2177-2185.

[148]BENGIO Y，DUCHARME R，VINCENT P，et al. A neural probabilistic language model[J]. Journal of Machine Learning Research，2003，3：1137-1155.

[149]GONZALEZ T F. ImageNet classification with deep convolutional neural networks[J]. Handbook of Approximation Algorithms and Metaheuristics，2007：1-1432.

[150]KIM Y. Convolutional neural networks for sentence classification[C]// Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha，Qatar：ACL，2014：1746-1751.

[151]WHITE M，VENDOME C，LINARES-VÁSQUEZ M，et al. Toward deep learning software repositories[C]// 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE，2015：334-345.

[152]NGUYEN A T，NGUYEN T N. Graph-based statistical language model for code[C]// 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE，2015：858-868.

[153]MOU L，LI G，ZHANG L，et al. Convolutional neural networks over tree structures for programming language processing[C]// Thirtieth AAAI Conference on Artificial Intelligence. 2016：1287-1293.

[154]MIKOLOV T，KARAFIÁT M，BURGET L，et al. Recurrent neural network based language model[C]// Eleventh Annual Conference of the International Speech Communication Association. New York：ACM，2010：1045-1048.

[155]HOCHREITER S，SCHMIDHUBER J. Long short-term memory[J]. Neural Computation，1997，9(8)：1735-1780.

[156]RAHMAN F，DEVANBU P. How，and why，process metrics are better[C]// 2013 35th International Conference on Software Engineering(ICSE). Piscataway，NJ：IEEE，2013：432-441.

[157]BETTENBURG N，PREMRAJ R，ZIMMERMANN T，et al. Extracting structural information from bug reports[C]// Proceedings of the 2008 International Working Conference on Mining Software Repositories. ACM，2008：27-30.

[158]POSHYVANYK D，GUÉHÉNEUC Y G，MARCUS A，et al. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval[J]. IEEE Transactions on Software Engineering，2007，33(6)：420-432.

[159]ZHANG Y，LO D，XIA X，et al. Fusing multi-abstraction vector space models for concern localization[J]. Empirical Software Engineering，2018，23(4)：2279-2322.

[160]PARNIN C，ORSO A. Are automated debugging techniques actually helping programmers? [C]// Proceedings of the 2011 International Symposium on Software Testing and Analysis. New York：ACM，2011：199-209.

NI Zhen，born in 1987. Ph.D candidate. Her main research interests include software data analysis，data-driven automated bug localization and software repair.

LI Bin，born in 1965，Ph.D，professor，Ph.D supervisor，is a senior member of China Computer Federation. His main research interests include cloud computing and intelligent software.

（责任编辑：何杨）