基于大数据的开源项目缺陷报告智能预检技术

席圣渠[1,2]，徐锋[1,2]，陈鑫[1,2]，李宣东[1,2]
1. 南京大学计算机科学与技术系，江苏 南京 210023；
2. 计算机软件新技术国家重点实验室（南京大学），江苏 南京 210023

摘要
缺陷报告预检目标在于确定优先级和修复措施，是保障软件可信的关键环节。然而，在日益普及的开源项目中，由于缺陷数量众多、缺乏组织管理等特性，人工预检难以及时完成，迫切需要基于大数据的自动化、智能化预检技术。结合工业界、学术界对缺陷报告预检的认知，提出了一种智能化缺陷报告预检技术框架，全面系统地归纳了缺陷报告预检中存在的3个关键任务：缺陷优先级分类、缺陷分派、缺陷再分派，并结合开源项目的特点提出了相关技术。实验结果初步验证了上述技术的合理性和有效性。

关键词
缺陷报告预检；缺陷优先级；缺陷分派；缺陷再分派

中图分类号：G434 文献标识码：A doi: 10.11959/j.issn.2096-0271.2021004

Big-data based intelligent bug triage techniques for open-source projects

XI Shengqu[1,2], XU Feng[1,2], CHEN Xin[1,2], LI Xuandong[1,2]
1. Department of Computer Science, Nanjing University, Nanjing 210023, China
2. State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing 210023, China

Abstract
Bug triage aims to determine the priority and repair measures and is critical in ensuring software trustability. However, in the increasingly popular open-source projects, due to a large number of defects and lack of organization and management, it is challenging to triage all the bug reports by hand on time, making big-data based, automated and intelligent bug triage urgent. An intelligent bug triage technical framework based on industry and academia's cognition was proposed, and three key tasks: bug priority classification, bug assignment, and bug reassignment, were identified comprehensively and systematically. Related technologies for the characteristics of open-source projects were proposed. The preliminary experiment results show the reasonableness and effectiveness of the above techniques.

Key words
bug triage, bug prioritization, bug assignment, bug re-assignment

1 引言

在软件开发过程中，软件测试与缺陷修复往往会占据软件总开发成本的三分之一以上[1]。为了更有效地管理缺陷，软件项目中普遍使用缺陷报告进行系统的记录和追踪。当缺陷报告被确认为新缺陷时，项目管理者首先会进行缺陷报告预检（以下简称缺陷预检），对缺陷报告进行预评估，以确定缺陷的优先等级、委派相关责任开发者进行修复，并在责任开发者无法按时完成修复时委派新的责任开发者，即缺陷再分派。缺陷预检位于缺陷报告生命周期的起点，良好的缺陷预检能极大地提高软件维护的效率与质量，是保障软件可信性的关键环节。

传统意义上，缺陷预检由项目管理者人工完成。然而，随着软件项目的不断扩大，缺陷的数量与日俱增，繁杂而耗时的缺陷预检给管理者带来了很大负担。对于快速发展，数量、种类、涵盖范围不断增大的开源软件项目而言，此类问题更是如此。由于开源项目开发者自由加入、流动性大、组织松散等特点，开源项目往往存在缺陷数量多、预检不及时、分派难度大等问题。据统计，截至2009年8月，开源项目Eclipse已收录超过25万份缺陷报告，开源项目Mozilla已收录超过50万份缺陷报告。平均而言，前者平均每天收到120份新缺陷报告，后者则有170份之多[2]。此外，提交到Eclipse项目的新缺陷报告平均需要16.7天才会得到项目管理团队的首次操作[3]。由此可见，在开源项目中，由于缺陷数量众多、组织管理松散、开发者自由加入且水平高低不一的特点，缺陷预检容易出现繁杂性、困难性、滞后性，因而需要自动化、智能化的方法辅助开源项目管理者进行缺陷预检，以保障开源软件的可信性，进而促进开源软件生态的良性发展。

另外，经过数十年积累，开源软件生态已经储备了大量与软件开发相关的数据，尤其是高质量的已修复缺陷报告，为自动化、智能化缺陷预检提供了先决条件。可通过数据挖掘、信息检索等方法，从大规模的已修复缺陷报告中学习相关的缺陷预检经验知识，用于辅助解决开源项目缺陷数量多、预检不及时、分派难度大的问题，从而提高开源项目软件维护的效率与质量，进而保障开源软件的可信性。软件项目从业者（管理者、开发者、测试者）普遍认可自动缺陷报告管理技术的积极作用[4]。具体到缺陷预检技术，可以利用已修复缺陷报告数据设计并训练相应机器学习模型，用于缺陷报告的优先级分类、为缺陷报告分派责任开发者，并在责任开发者无法及时完成修复任务时，推荐新的责任开发者。

上述基于软件大数据的智能化缺陷预检技术已成为当前的研究热点[3-31]，然而，当前研究的主要关注点在于如何利用新的机器学习模型与方法、特征提取技术，或寻求更多的数据特征，以取得在测试数据集上的准确率的提升。当前技术较少考虑开源项目的特点，难以满足开源项目缺陷预检的需求。

为此，本文首先根据开源项目的特点，归纳了缺陷预检的3个关键任务及其挑战，并提出一套完整的自动化缺陷预检框架与技术。

2 开源项目缺陷预检的关键任务与挑战

软件缺陷管理一直是软件维护实践中的重要内容，也是软件工程研究的重点问

题。从工业界和学术界的普遍认知来看，缺陷预检是缺陷报告生命周期（如**图1**所示）的开端和重要环节，缺陷预检包含以下3个较为关键的任务。

● 缺陷优先级分类：用于确定缺陷修复的优先次序，对应“新缺陷”向“已分派缺陷”转换的过程，人工进行缺陷优先级分类往往要求缺陷报告管理者具有丰富的经验和对软件项目当前目标的充分理解，而错误的优先级分类将导致项目团队资源利用率的低下[5]。

● 缺陷分派：用于确定新缺陷报告的责任开发者，对应的缺陷报告将被标记为“已分派缺陷”，人工进行缺陷分派往往要求缺陷报告管理者逐条阅读缺陷报告，并将其与合适的开发者匹配，而错误的缺陷分派会导致缺陷再分派，进而延误缺陷修复进程[3]。

● 缺陷再分派：当责任开发者无法修复缺陷时，选择新的责任开发者。对于已分派缺陷、归档缺陷，由于责任开发者能力不足、无法及时处理等原因，需要进行缺陷再分派。不及时的缺陷再分派会影响缺陷修复效率，而开源项目中有接近40%的缺陷报告需要至少一次再分派[3]。

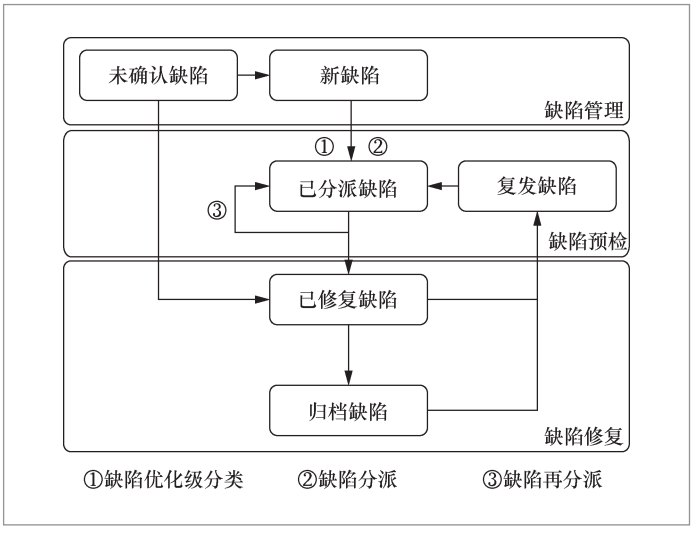

图1 缺陷报告生命周期

开源项目组织管理松散、开发者可自由加入且水平高低不一等特点使得缺陷预检中上述3个关键任务的自动化面临挑战，具体如下。

● 缺陷优先级分类方面。开源项目管理松散，缺乏专职的缺陷管理者，这使得开源项目中存在大量未分类的缺陷报告（其优先级为默认优先级），导致从开源项目中获得的训练数据的优先级分布很不均衡，即默认优先级缺陷报告数量巨大，而其他优先级类别的缺陷报告数量较少。若将其简单建模为分类问题，则容易出现默认优先级分类预测准确率较高，而真正影响项目团队资源分配的边沿优先级（edge priority），即较低或较高优先级的分类预测准确率低下的情况。因此，如何利用开源项目中的分类标签不平衡且存在大量噪声的训练数据来提高边沿优先级的预测准确率，成为缺陷优先级分类任务面临的主要挑战。

● 缺陷分派方面。由于开源项目开发者可自由加入，大量开发者利用业余时间加入开源项目，不同开发者的参与意愿强弱存在差异，表现为投入项目的时间长短不一，活跃程度也有差别。若简单地推荐能力与缺陷描述相近的开发者，容易出现开发者失效，即开发者长时间无响应，甚至开发者已经离开项目的情况，严重时可影响缺陷修复流程[6]，降低缺陷分派准确率；相反，在存在多名能力相近的开发者时，选择参与意愿强、活跃程度高的开发者更符合现实场景需求。因此，如何动态地建模和评估开发者的参与意愿，并在能力相近的开发者中优先推荐活跃程度高的开发者，以降低失效风险、提高缺陷分派准确率，成为缺陷分派任务面临的主要挑战。

● 缺陷再分派方面。由于开源项目开发者水平高低不一，当开发者因能力不足等原因无法修复缺陷时，往往需要寻找经验更丰富且原有责任开发者信赖的开发者，以完成缺陷的修复。若将问题简化为缺陷分派，则缺乏对原有责任开发者无法修复缺陷这一反馈信息的利用，会降低缺陷再分派的准确率。另外，用作训练的已修复缺陷报告中往往包含大量缺陷传递序列，即责任开发者变更的历史记录，其中隐含着开发者之间经验与信赖的偏序关系，其为反馈信息的利用提供了可参照的经验规律。因此，如何充分使用缺陷传递序列、挖掘其中隐含的开发者之间的经验与信赖关系，以利用无法修复的反馈信息来提升缺陷再分派的准确率，成为缺陷再分派任务面临的主要挑战。

3 相关工作

缺陷预检领域存在大量缺陷优先级分类[7-15]、缺陷分派[16-27]的研究工作，以及源于或服务于工业界的文献资料[2,21]。这些工作普遍基于缺陷报告文本信息，采用机器学习方法构建分类问题，或采用信息检索方法构建相似度模型。而对于开源项目中较为突出的缺陷再分派问题，相关研究工作较少。因此，本节将主要介绍前两者的研究思路与进展，并讨论其在应对开源项目预检任务面临的挑战时存在的不足。

3.1 缺陷优先级分类的相关工作

参考文献[7]最早提及缺陷报告优先级分类，采用监督与无监督混合的方法将相似缺陷报告聚类，并认为同一个聚类内的缺陷报告描述相似或相同的问题，因而应当具有相同的优先级；作者在特征选取、聚类方法、可视化等方面做了较为深入的研究，然而该方法未对缺陷报告优先级做出明确判断，较难进行实际应用。

随后，Kanwal J等人[8]提出基于支持向量机（support vector machine，SVM）的分类方法，目标是为新缺陷报告分配一个优先级级别，优先级高的缺陷报告将更受项目组重视，并更快被修复。该方法将精度（precision）和召回率（recall）作为评估指标，并通过实验验证了方法的有效性。Tian Y等人[9]深入研究了多种可能影响优先级分类的因素，以选取更有效的训练特征，并采用线性回归和阈值截断的方法训练分类模型，以缓解训练数据不平衡的问题。Umer Q等人[10]进一步研究了情感分析在缺陷优先级分类中的作用，采用支持向量机模型，将缺陷报告情感特征作为输入，以提升优先级分类的准确率。其后，Umer Q等人[11]进一步研究了分类器的影响，采用基于深度学习的分类模型，用卷积神经网络（convolutional neural network，CNN）抽取文本特征，并融合情感特征，共同预测缺陷优先级。然而，这些工作仍存在边沿优先级预测准确率低的问题，难以应用于实际软件的开发过程，需要进一步研究优化改进的相关技术。

另外，区别于以上预测明确优先等级的工作，Valdivia-Garcia H等人[12]关注检测阻塞缺陷，此类缺陷会影响其他缺陷的修复，因而具有很高的修复优先级。作者基于6个大型开源项目数据，总结出了与阻塞缺陷关联较为密切的14个因素，并以此建立了决策树，以辨别阻塞缺陷。Jiang Y等人[13]则将缺陷报告按严重程度分为2类，并采用排序学习（learning to rank）过滤掉部分严重程度低的缺陷报告，以建立更有效的数据集，最终采用监督学习进行训练和预测。Yang X L等人[14]关注高影响力缺陷报告的识别，结合非均衡学习策略

（imbalanced learning strategies），缓解数据不均衡的问题。Ren H等人[15]则对阻塞缺陷进行了实证研究，从修复时间、开发者经验等5个方面开展调研，结果表明，阻塞缺陷的修复往往更加耗时耗力，应当具有更高的修复优先级。然而，此类工作往往将缺陷报告简化为二分类问题，缺乏对缺陷优先级的完整辨别与评估，降低了优先级分类的精确性。

综上所述，相关工作或者边沿优先级预测准确率低，难以应用于实际的开源软件项目开发过程；或者将缺陷报告简化为二分类问题，降低了优先级分类的精度，难以为开源项目的缺陷修复决策提供明确的信息。

3.2 缺陷分派的相关工作

缺陷分派问题最早由Cubranic D等人[16]提出，作者将问题建模为文本分类（开发者被视为类别）任务，输入为缺陷报告文本，输出为缺陷报告的候选责任开发者；利用标记数据训练朴素贝叶斯模型，在开源项目数据集上验证了方法的有效性。随后，Anvik J等人[17]针对开源项目中的缺陷分派问题做了系统性的调研，并在朴素贝叶斯模型的基础上，使用支持向量机、决策树等方法来解决此类问题，结果显示支持向量机的方法取得了最好的效果。由于开源项目具有开发周期长、开发者流动性大的特点，Tamrawi A等人[18]提出了基于模糊集的方法——Bugzie方法，通过过滤不活跃的开发者来提高预测准确率；然而，过滤筛选的方式过于简单，无法刻画开发者活跃程度的差异。

在文本信息之外，研究者也尝试结合缺陷报告隶属的产品、模块等元数据信息。Yang G等人[19]首先使用主题模型抽取缺陷报告文本的特征，随后在元数据一致的限制下查找相似缺陷报告，最后基于这些相似缺陷报告，推荐责任开发者。Xia X等人[20]则将元数据作为监督信息，设计全新的主题模型结构，以获得更好的文本特征表示，并以此对开发者能力特征进行建模，基于新缺陷报告与开发者能力特征的匹配程度来推荐责任开发者。然而，虽然主题模型（如LDA[32-33]）在自然语言处理方面得到了广泛应用，但仍存在无法对词序关系进行建模等问题，在文本内容理解上仍有提升空间。

近期，随着深度学习方法在自然语言处理领域的广泛应用，也出现了基于深度学习的缺陷分派研究。Lee S R等人[21]使用卷积神经网络建模缺陷报告文本，并在多个工业界项目上验证了方法的有效性。Mani S等人[22]则使用基于注意力机制的双向循环神经网络建模缺陷报告文本，以求准确率的提升。然而，这些方法大部分局限于缺陷报告文本，未能充分使用缺陷报告蕴含的多种信息。

研究者也尝试融入代码、相似缺陷报告、修复记录等信息，进一步提升缺陷分派准确率。Alkhazi B等人[23]结合代码相似度，认为编写过缺陷相关代码的开发者更适合修复缺陷。Zhang T等人[24]发现较短的缺陷报告不利于分派，故通过查找相似缺陷报告的方式拓展缺陷报告文本，并在此基础上推荐责任开发者。Sun X B等人[25]关注安全缺陷，并对开发者的安全缺陷修复经验进行建模，针对严重的安全缺陷，推荐经验丰富的开发者。Yadav A等人[26]对开发者修复效率进行建模，并采用相似度的方式筛选责任开发者。Guo S等人[27]则关注开发者的修复历史，优先选取近期有相似修复经验的开发者。然而，这些方法普遍采用较简单的语言模型，难以精确刻画缺陷报告的文本特征。

综上所述，相关工作存在信息使用不充分、文本建模较简单的问题，导致开发者能力刻画不全面、匹配缺陷相关开发者

效果较差，且缺乏对开源项目中开发者活跃度的差异化区分，难以在能力相近的开发者中识别最活跃的开发者。

3.3 缺陷再分派的相关工作

缺陷再分派方面的研究工作最近才被人们重视，相关研究工作较少，较为相似的工作如下所述。

Jeong G等人[3]整合缺陷再分派过程中责任开发者传递、变更的有向图，并以此为依据，调整缺陷分派结果，提高传递图上位于终点的开发者的排名，以此优化缺陷分派效果。Bhattacharya P等人[28]则在Jeong G等人[3]的基础上进一步优化，通过分析包含元数据的传递图，进一步优化缺陷分派效果。Wu H R等人[29]对缺陷传递序列的长度进行了研究，发现产品、模块等4项属性对缺陷传递序列的长度有显著影响。然而，这些工作仍停留在对缺陷分派问题的优化层面，缺乏对缺陷再分派任务的关注。

Xia X等人[30]则研究了复发缺陷的成因，并提出了基于集成学习的复发缺陷预测方法。然而，该方法仍只关注复发缺陷的预测，缺乏复发缺陷可能存在的责任开发者变更的解决方法。Mi Q等人[31]发现，复发缺陷中存在部分不影响用户体验的非负面（non-negative）缺陷，此类缺陷普遍较容易修复，且不需要变更责任开发者。然而，这些工作更关注调研、分析层面，缺乏具体的应对缺陷再分派任务的方法。

综上所述，缺陷再分派方面的研究工作较少，难以简单采用缺陷分派技术应对开源项目中存在的大量缺陷再分派问题。

4 缺陷报告智能预检技术框架

为了系统地解决开源软件项目缺陷预检中存在的3个关键任务，本文提出了一个基于大数据资源的缺陷报告智能预检技术框架，如**图2**所示。框架以开源项目缺陷追踪系统中海量已修复缺陷报告的大数据资源为数据支撑。针对新缺陷，提出基于半监督学习的缺陷优先级分类方法、基于开发者活跃度的缺陷分派方法，分别预测缺陷报告的优先级和责任开发者；针对已分派缺陷或复发缺陷，提出基于开发者信赖关系的缺陷再分派方法，预测新的责任开发者。框架主要包括以下几种方法。

- 基于半监督学习的缺陷优先级分类方法：由于开源项目组织管理松散，容易出现优先级未分类的情况，导致默认优先级P3中混杂了更高、更低优先级的数据，可信性不足。另外，由于P3数据中存在大量真实优先级标签，全部丢弃也将影响训练结果。假设P3中的数据符合以P3为均值的正态分布，结合其文本对标签进行拟合，扰动得到P3数据的可能的优先级分布；将扰动后的数据作为训练数据，在牺牲部分P3准确率的代价下，提升边沿优先级的准确率。因此，为了充分利用已有数据，使用半监督学习的方法来提升缺陷边沿优先级的分类准确率。
- 基于开发者活跃度的缺陷分派方法：由于开源项目开发周期长、开发者自由加入，开发者存在数量巨大、人员流动频繁、活跃周期不固定的特点，优先推荐活跃程度高的开发者有利于又快又好地修复缺陷。为评估当前开发者的活跃程度，可以参照开发者在缺陷追踪系统中的活动记录，评估与缺陷报告相关度较高的开发者，当前时刻附近活动记录越多的开发者越活跃，距离当前时刻越近的活动对活跃程度的影响越大。因此，结合开发者的活动记录，对开发者活跃程度进行度量评估，以提升缺陷分派的准确率。
- 基于开发者信赖关系的缺陷再分派方法：由于开源项目的开发者水平高低不

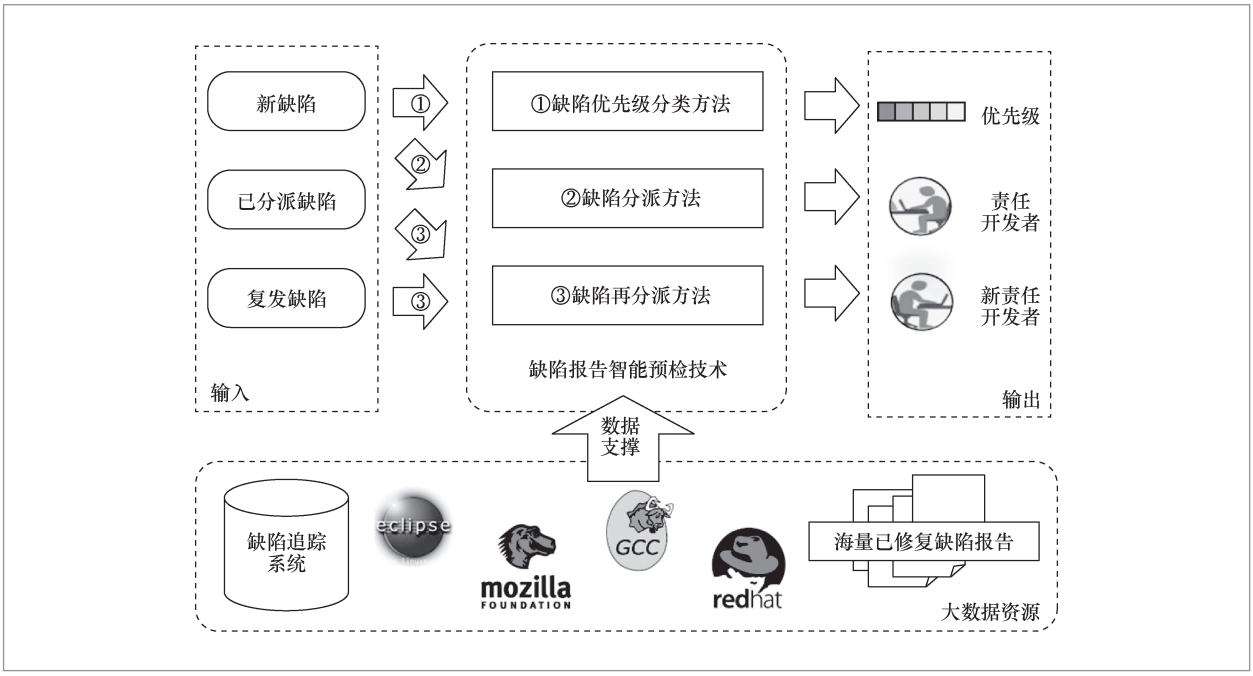

图 2 缺陷报告智能预检技术框架

一，责任开发者无法完成修复时，往往要求助于能力更强且自身信赖的开发者。为了充分利用责任开发者无法完成修复这一反馈信息，可参照缺陷传递序列，对开发者之间的信赖关系进行建模，靠近序列末尾的开发者往往具有更加丰富的经验，且更受其他开发者信赖。另外，缺陷的提交者也可能是开发者，将提交者纳入信赖关系的建模有助于提高建模准确性。因此，为充分利用反馈信息，需结合缺陷传递序列，对开发者信赖关系进行建模，以提升缺陷再分派的准确率。

后文将依次对上述3个方法与技术进行具体介绍。

4.1 基于半监督学习的缺陷优先级分类方法

缺陷优先级分类的目标是预测新缺陷报告的优先等级。由于开源项目管理松散，存在大量默认优先级的缺陷报告，数据分布很不均衡[9]，导致相关工作普遍出现分类失衡的状况[9-10]。因此，采用半监督学习的方法将默认优先级数据视为无标签或标签可信性不高的数据，在尽可能牺牲少量默认优先级精度的情况下，换取边沿优先级的准确性。为此，笔者提出了一个基于生成对抗网络[34-35]的半监督学习方法——BrCCGAN来解决上述问题。

该方法的主要思想是：给定缺陷报告br，需训练评分函数S(br)=prior∈{P1, P2, P3, P4, P5}。其中，边沿优先级P1、P2、P4、P5更重要；P3为默认优先级，其标签可信性不高[36]。因此，通过扰动默认优先级P3的标签，拟合出符合文本的P3数据的标签分布，以损失部分P3分类准确率为代价，提高边沿优先级的分类准确率。

基于半监督学习的缺陷优先级分类方法的工作流程如**图3**所示。具体而言：①将缺陷报告br和优先级prior视为完整的数

据对<br,prior>，优先级准确的数据对具有较高的融洽度，而优先级不准确的数据对融洽度较低；②生成器通过符合高斯分布的随机扰动*z*来变更数据对中的优先级，产生虚假的优先级prior'；③判别器检验缺陷报告br与真实优先级prior或虚假优先级prior'的融洽度；④当生成器与判别器互相博弈，达到稳定收敛状态时，固定其内部缺陷报告br的特征抽取网络，并用此网络抽取的特征单独训练分类模型，即最终的评分函数*S*。

此方法中生成器、判别器相互博弈的过程可表示为：

prior'= G(br, *z*)

$$Objective = \min_G \max_D E[\log D(br, prior)] + E[\log(1 - D(br, prior'))] \quad (1)$$

其中，G表示生成器，其输入为缺陷报告br和符合高斯分布的随机扰动*z*，输出为虚假的优先级prior'；D表示判别器，其输入为缺陷报告br与优先级组成的数据对，优先级可能为真实优先级prior或虚假优先级prior'；Objective表示目标函数，*E*表示数学期望，训练过程需依次最小化生成器G的损失、最大化判别器D的损失，以达到两者相互博弈的目的。

整个模型包含训练、预测两个部分。训练部分，将由优先级确定的缺陷报告大数据作为输入，此过程的主要目标为训练生成器G和判别器D，为了降低参数规模，将两者缺陷报告特征抽取网络的参数共享，以简化训练过程、降低训练开销。预测部分，使用缺陷报告特征抽取网络抽取新缺陷报告的特征向量，并使用独立的优先级分类模型预测优先级prior。

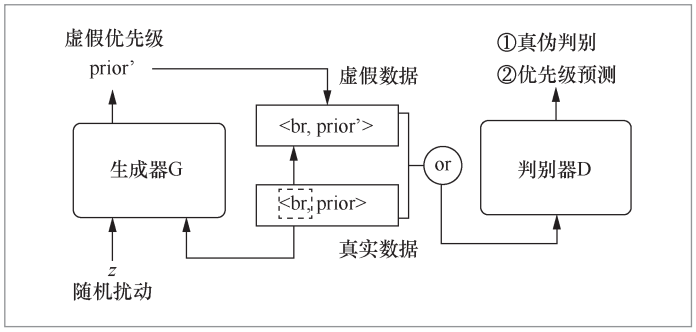

图3 基于半监督学习的缺陷优先级分类方法的工作流程

笔者基于CCGAN[34]实现了上述优先级分类方法BrCCGAN，主要改进在于选择了更合理的文本特征抽取方法，并优化了整个生成对抗网络的结构。

为了验证方法的有效性，选取开源项目Eclipse作为实验对象，共收集166 081份缺陷报告，按7:1:2的比率将其分割为训练集、验证集和测试集；选取基于线性回归和阈值截断的DRONE[9]、基于双向循环神经网络的BiRNN[10]、基于卷积神经网络和情感分析的CPur[11]作为对比方法；选取精度和召回率作为评估指标。

基于半监督学习的缺陷优先级分类方法BrCCGAN的实验结果见**表1**，横向为不

表1 基于半监督学习的缺陷优先级预测方法的精度与召回率

对比方法	P1		P2		P3		P4		P5	
	精度	召回率	精度	召回率	精度	召回率	精度	召回率	精度	召回率
DRONE	34.51%	32.58%	15.83%	21.68%	86.89%	80.49%	11.98%	16.91%	5.00%	16.28%
BiRNN	38.99%	36.46%	21.31%	23.41%	88.93%	88.45%	13.59%	18.38%	8.47%	17.44%
CPur	39.88%	37.22%	20.54%	22.77%	89.72%	88.57%	12.44%	18.38%	7.94%	18.60%
BrCCGAN	42.65%	41.64%	24.79%	24.77%	86.75%	82.12%	24.32%	26.47%	17.59%	22.09%

同优先级下的精度与召回率，纵向为选取的对比方法。实验结果显示，较同类工作，BrCCGAN的P3优先级预测准确率稍有下降，但边沿优先级的预测准确率有显著提升，表明本文所提优先级分类方法具有较强的实用价值。

4.2 基于开发者活跃度的缺陷分派方法

缺陷分派的目标是推荐责任开发者。开源项目容易出现推荐的责任开发者失效的情况。另外，当前较为活跃的开发者往往愿意为项目贡献更多的力量。因此，笔者提出了一个基于循环神经网络的开发者活跃度建模方法DeepTriage来进行缺陷分派。

本方法的主要思想是：在利用循环神经网络建模缺陷报告文本词序关系的同时，对开发者活跃度进行建模，优先推荐当前活跃的开发者修复缺陷，以缓解推荐的开发者无法修复、修复效率低的问题，即给定缺陷报告br、修复记录信息his，需训练评分函数$S(\text{br},\text{his})=d\in D$。其中，$D$表示开源项目开发者的集合。

基于开发者活跃度的缺陷分派方法的工作流程如**图4**所示。具体而言，包括3个重要部分：①文本高层特征抽取，对缺陷报告文本br建模，并获取文本特征表示ht；②开发者活跃度高层特征抽取，建模修复记录信息his，并获取活跃度特征表示ha；③高层特征融合，将ht和ha的特征表示进行融合，以获取分类器的输入特征。以下将依次对这3个部分进行详细介绍。

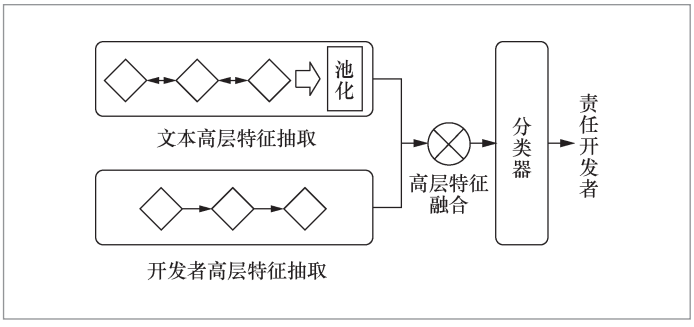

图 4 基于开发者活跃度的缺陷分派方法的工作流程

（1）文本高层特征抽取

文本高层特征抽取使用了双向循环神经网络。得益于循环神经网络在文本分类上的进步[37-38]，基于双向循环神经网络的文本高层特征抽取方法逐渐成为主流。具体而言，以独热编码的形式依次输入单词，从顺序、倒序两个方向进行遍历；对于每个单词，将其正向、逆向的特征进行拼接，输入最大池化层，以整合单词特征，得到保留了最关键信息的整体表示，此过程可表示为：

$$\text{ht}_i=[\text{forward}(\text{ht}^+_{i-1},t_i),\text{backward}(\text{ht}^-_{i+1},t_i)]$$

$$\text{ht}=\text{maxpooling}(\text{ht}_1,\text{ht}_2,\cdots,\text{ht}_N) \quad (2)$$

其中，$\text{br}=(t_1,t_2,\cdots,t_N)$表示缺陷报告的文本信息，第$i$个单词表示为$t_i$；$N$表示文本长度；forward、backward分别表示双向循环神经网络的正向、逆向遍历过程；ht_i表示第i个单词的高层特征，ht^+_{i-1}和ht^-_{i+1}分别表示第$i-1$个单词的正向特征和第$i+1$个单词的逆向特征；maxpooling表示最大池化层；ht表示缺陷报告的文本高层特征。

（2）开发者活跃度高层特征抽取

开发者活跃度高层特征抽取首先获取开发者的修复记录信息his，然后将修复记录信息输入神经网络，以得到高层特征。在修复记录获取方面，给定新缺陷报告的提交时刻，需获取提交时刻向前一段时间内的修复记录，并将其按照时间顺序排列；由于开源项目缺陷报告数量巨大，为了约束his的长度，采用目标缺陷报告的产品、模块元数据进行过滤，即只保留产品、模块相同的修复记录；若无法找到相应修复记录，则his被设置为空列表。在高层特征抽取方面，采用单向循环神经网络获得最后节点的状态，以保障越靠近当前时间节点

的修复记录影像越大，此过程可表示为：

$$ha = forward(ha_{M-1}, his_M) \quad (3)$$

其中，forward表示顺序输入的单向循环神经网络；M表示修复记录的长度；ha_{M-1}表示第$M-1$个修复记录对应的高层特征，其值同样由forward输出得到；his_M表示第M条修复记录；ha表示开发者活跃度高层特征表示，由于将最后节点的状态作为特征表示，故$ha=ha_M$。

（3）高层特征融合

高层特征融合将文本高层特征、开发者活跃度高层特征融合在一起，共同进行后续预测，即：

$$h = ht \oplus ha \quad (4)$$

其中，ht和ha分别表示文本高层特征、开发者活跃度高层特征；h表示融合后的特征；⊕表示特征融合操作，是此步骤的核心。特征融合操作通常采用拼接、元素间相加、元素间相乘3种较为简单的方法。本文采用特征向量相互影响更为激烈的元素间相乘方法，以便让开发者活跃度特征更多地影响责任开发者的预测。

模型的最后，评分函数S需建立融合的高层特征到项目开发者的映射，并选择预测值最高的开发者作为推荐的责任开发者。采用深度学习中常用的全连接层完成映射，可表示为：

$$d = argmax(dense(h)) \quad (5)$$

其中，h表示融合后的高层特征，dense为全连接层，其输出维度等同于项目中开发者的数量$|D|$；argmax表示选择预测值最高的开发者；d表示推荐的责任开发者的编号。

整个模型包含训练、预测两个部分。训练部分，将已经标记归档的缺陷报告大数据作为输入，这些缺陷报告皆包含真实修复缺陷的责任开发者信息，并可抽取出文本信息及开发者修复记录信息。这些数据共同构成训练实体，完成模型的训练。测试部分，输入新的缺陷报告，以及从缺陷追踪系统中抽取的开发者活动记录，则可得到推荐的责任开发者d。

笔者基于oh-LSTMp（one-hot LSTM with pooling）[37]实现了上述缺陷分派方法DeepTriage，主要改进在于增加了开发者活跃度的特征抽取网络，并试验了多种特征融合方法对分派结果的影响。

为了验证方法的有效性，选取开源项目Eclipse、Netbeans、Mozilla、Redhat作为实验对象，将缺陷报告按照时间排序，并分为11份，每次选取前n份作为训练集、第$n+1$份作为测试集，共进行10轮实验，并对比平均结果；选取经典分类方法SVM[39]、Yang等人[19]基于主题模型并融合元数据的方法提出的模型（以下简写为Yang）、TopicMinerMTM[20]作为对比方法；选取准确率（accuracy）作为评估指标。

基于开发者活跃度的缺陷分派方法DeepTriage的Top-1准确率见**表2**，纵向表示选取的4个实验对象，横向表示选取的对比方法。实验结果显示，与对比算法相比，DeepTriage的Top-1准确率在各个

表2 基于开发者活跃度的缺陷分派方法的Top-1准确率

实验对象	SVM	Yang	TopicMinerMTM	DeepTriage
Eclipse	28.05%	32.12%	36.89%	41.66%
Netbeans	29.32%	35.29%	42.37%	49.30%
Mozilla	20.43%	26.06%	29.04%	32.32%
Redhat	22.76%	39.83%	45.91%	48.58%

项目中均有显著提升，表明优先推荐活跃度高的开发者有利于切实完成缺陷修复工作，本文所提缺陷分派方法具有较强的实用价值。

4.3 基于开发者信赖关系的缺陷再分派方法

缺陷再分派的目标是在需要变更责任开发者时，推荐新的责任开发者。不同于缺陷分派任务，缺陷再分派包含额外的传递序列tos信息，tos中靠后的开发者通常具有更强的技术水平，更加受到其他开发者信赖；建模开发者之间的信赖关系有利于缺陷的有效修复，且更加符合项目开发者的再分派任务。为此，笔者提出了一个基于编码器、解码器（encoder-decoder）模型[40-41]的方法iTriage来解决上述问题，以同时建模缺陷报告文本信息与开发者之间的信赖关系。

本方法的主要思想是：将传递序列tos建模为解码过程，预训练开发者间的信赖关系；当需要进行缺陷再分派时，使用此预训练模型获取文本、传递序列的特征，并结合元数据特征共同预测最终的责任开发者。此过程避免了对缺陷传递过程的模拟，直接关注最终目标，有利于提高缺陷再分派的准确率，即给定需要变更责任开发者的缺陷报告br、缺陷报告的传递序列tos，需训练评分函数S，使得S(br,tos)=$d\in D$。其中，D表示开源项目开发者的集合；tos表示从缺陷提交者开始，包含一个或多个责任开发者的传递序列。对于已修复的缺陷，tos中最后的开发者表示完成修复的开发者；对于需再分派的缺陷，则表示无法完成修复的开发者，通常需要指定不同的新责任开发者。

基于开发者信赖关系的缺陷再分派方法的工作流程如**图5**所示。具体而言，包括3个重要部分：①文本编码器，建模缺陷报告文本br的特征；②传递序列解码器，模拟缺陷报告的传递过程，通过预训练的方式学习开发者间的信赖关系；③开发者推荐模型，结合文本特征、传递状态特征和元数据特征，共同推荐新的责任开发者。以下将依次对这3个部分进行详细介绍。

（1）文本编码器

文本编码器采用双向循环神经网络建模缺陷报告文本特征。区别于缺陷分派，此处直接输出每个单词的特征表示，并记录遍历所有单词后的状态特征，可表示为：

$$(\text{state}_0, [\text{ht}_1,\text{ht}_2,\cdots,\text{ht}_N]) = \text{encoder}(\text{br}) \quad (6)$$

其中，br=(t_1,t_2,$\cdots$,t_N)表示缺陷报告文本的集合；ht_i表示第i个单词的高层特征，这些特征将通过后续的注意力机制进行整合；state_0表示完整遍历后的状态信息，其中数字0表示解码器的初始状态。

（2）传递序列解码器

传递序列解码器采用循环神经网络建模。解码器通过循环神经网络模拟缺陷报告在责任开发者间传递的过程，学习开发者之间的信赖关系。为确定j时刻输出的开发者，需要缺陷报告的全部文本特征、$j-1$时刻的隐含状态、$j-1$时刻的责任开发者。可表示为：

$$(\text{state}_j, d_j) = \text{decoder}(\text{state}_{j-1}, [\text{ht}_1, \text{ht}_2,\cdots, \text{ht}_N], d_{j-1}) \quad (7)$$

其中，state_j、d_j分别表示j时刻的隐含状态、输出的责任开发者。state_j的初始状态为解码器最后输出的状态；d_j的初始状态则区别于传统模型采用特殊开始符<start>的方法，本节将缺陷报告提交者作为初始状态输入，由于存在提交者也是开发者的情况，加入缺陷报告提交者能够更有效地建立信赖模型。此过程将一直持续，直到遇到代表序列结束的特殊结束符<end>。

在此过程中，也将采用注意力机制整

合文本特征。具体方法为：使用解码器$j-1$时刻的隐含状态，重新计算单词特征的权重，并加权求和。可表示为：

$$c_j=\text{attention}(\text{state}_{j-1}, [\text{ht}_1,\text{ht}_2,\cdots,\text{ht}_N]) \quad (8)$$

其中，c_j表示j时刻的完整缺陷报告文本上下文特征。

（3）开发者推荐模型

开发者推荐模型采用基于全连接层的分类方法。其输入包括当前j时刻的文本上下文特征、解码器隐含状态特征，以及缺陷报告整体的元数据特征。可表示为：

$$d=\text{recommend}(c_j,\text{state}_j,m) \quad (9)$$

其中，m表示元数据特征，此特征可使用简单的嵌入层获得；由于元数据在缺陷报告中长期保持稳定，因而在任意时刻j都可使用相同的元数据特征m。

整个模型包括训练、预测两个部分。训练部分，将归档缺陷报告大数据作为监督，学习并优化模型参数。测试部分，模型参数固定，给定待变更责任开发者的缺陷报告，首先经过文本编码器、传递序列解码器获取文本上下文特征c、解码器隐含状态特征state；其次，提取元数据，并得到元数据特征m；最后，将3种特征输入开发者推荐模型，则可得到推荐的新责任开发者。

笔者基于编码器、解码器模型[41]实现了上述缺陷再分派方法iTriage，主要改进在于将缺陷在开发者间传递的过程建模为解码过程，并添加了开发者推荐模型，结合缺陷再分派时刻的文本上下文特征、传递状态特征、元数据特征，直接预测最终的修复者。

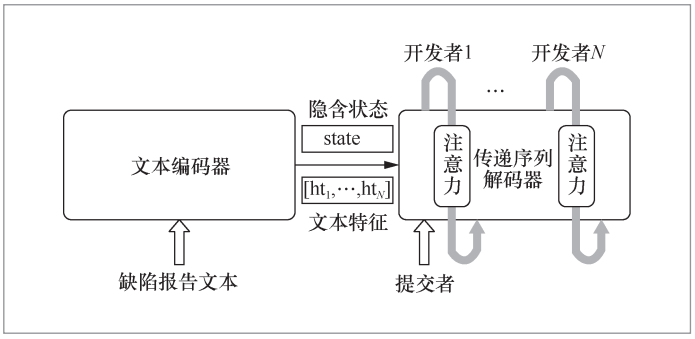

图 5 基于开发者信赖关系的缺陷再分派方法的工作流程

为了验证方法的有效性，选取开源项目Eclipse、Mozilla、Gentoo作为实验对象，采用模拟时序的十折验证作为测试方法；选取经典文本分类方法SVM[39]、基于注意力机制的循环神经网络DBRNN+A方法[21]、基于卷积神经网络的方法CNN Triager[22]作为对比方法；选取准确率作为评估指标。

基于开发者信赖关系的缺陷再分派方法iTriage的Top-1准确率见**表3**，纵向表示实验对象，横向表示对比方法。此外，实验进一步验证了缺陷再分派情况下的分派准确率，iTriage@1表示第一次分派的预测准确率，iTriage@2、iTriage@3分别表示第一次、第二次再分派（即第二次、第三次分派）的准确率。实验结果显示，iTriage第一次分派的预测准确率已有显著提升，表明了将提交者纳入信赖关系建模的有效性；且在随后的第一次、第二次再分派中，准确率得到进一步提升，表明了缺

表 3 基于开发者信赖关系的缺陷再分派方法的 Top-1 准确率

实验对象	SVM	DBRNN+A	CNN Triager	iTriage@1	iTriage@2	iTriage@3
Eclipse	28.06%	28.71%	28.92%	47.86%	54.15%	62.33%
Mozilla	20.43%	20.74%	20.83%	37.31%	42.31%	44.68%
Gentoo	22.76%	24.71%	24.99%	48.58%	56.33%	58.58%

陷报告在开发者间传递的反馈信息的有效性，本文所提缺陷再分派方法具有较强的实用价值。

5 结束语

本文提出了一个智能化缺陷报告预检技术的框架，结合开源项目的特点，并利用已修复缺陷报告的大数据作为支撑，基于深度学习的相关模型与技术，针对缺陷预检的3个关键任务提出了相应的自动化技术：基于半监督学习的缺陷报告优先级预测方法、基于开发者活跃度的缺陷分派方法、基于开发者信赖关系的缺陷再分派方法。

面向开源项目的缺陷报告预检技术，尤其是缺陷再分派技术的研究日益被重视，仍有许多问题需要更加深入的分析和研究，如缺陷报告质量对缺陷预检工作的影响如何，代码托管平台中缺陷报告和代码变更的关联信息、开发者评价信息等如何用于提高缺陷预检的准确率等。

参考文献:

[1] XIE T, ZHANG L, XIAO X, et al. Cooperative software testing and analysis: advances and challenges[J]. Journal of Computer Science & Technology, 2014, 29(4): 713–723.

[2] GUO P J, ZIMMERMANN T, NAGAPPAN N, et al. Characterizing and predicting which bugs get fixed: an empirical study of Microsoft Windows[C]//The 32nd ACM/IEEE International Conference on Software Engineering. [S.l.:s.n.], 2010: 495–504.

[3] JEONG G, KIM S, ZIMMERMANN T. Improving bug triage with bug tossing graphs[C]//The 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. New York: ACM Press, 2009: 111–120.

[4] ZOU W, LO D, CHEN Z, et al. How practitioners perceive automated bug report management techniques[J]. IEEE Transactions on Software Engineering, 2020, 46: 836–862.

[5] ALENEZI M, BANITAAN S. Bug reports prioritization: which features and classifier to use?[C]//2013 12th International Conference on Machine Learning and Applications. Piscataway: IEEE Press, 2013: 112–116.

[6] SAHA R K, KHURSHID S, PERRY D E. Understanding the triaging and fixing processes of long lived bugs[J]. Information and Software Technology, 2015, 65(C).

[7] PODGURSKI A, LEON D, FRANCIS P, et al. Automated support for classifying software failure reports[C]//The 25th International Conference on Software Engineering. Piscataway: IEEE Press, 2003: 465–475.

[8] KANWAL J, MAQBOOL O. Bug prioritization to facilitate bug report triage[J]. Journal of Computer Science and Technology, 2012, 27(2): 397–412.

[9] TIAN Y, LO D, SUN C. Drone: predicting priority of reported bugs by multi-factor analysis[C]//2013 IEEE International Conference on Software Maintenance. Piscataway: IEEE Press, 2013: 200–209.

[10] UMER Q, LIU H, SULTAN Y. Emotion based automated priority prediction for bug reports[J]. IEEE Access, 2018, 6: 35743–35752.

[11] UMER Q, LIU H, ILLAHI I. CNN-based automatic prioritization of bug reports[J]. IEEE Transactions on Reliability, 2019(99): 1–14.

[12] VALDIVIA-GARCIA H, SHIHAB E.

Characterizing and predicting blocking bugs in open source projects[C]//The 11th Working Conference on Mining Software Repositories. New York: ACM Press, 2014: 72–81.

[13] JIANG Y, LU P, SU X, et al. LTRWES: a new framework for security bug report detection[J]. Information and Software Technology, 2020, 124: 106314.

[14] YANG X L, LO D, XIA X, et al. High-impact bug report identification with imbalanced learning strategies[J]. Journal of Computer Science and Technology, 2017, 32(1): 181–198.

[15] REN H, LI Y, CHEN L. An empirical study on critical blocking bugs[C]//The 28th International Conference on Program Comprehension. New York: ACM Press, 2020.

[16] MURPHY G, CUBRANIC D. Automatic bug triage using text categorization[C]//The 16th International Conference on Software Engineering & Knowledge Engineering. [S.l.:s.n.], 2004: 1–6.

[17] ANVIK J, HIEW L, MURPHY G C. Who should fix this bug?[C]//The 28th International Conference on Software Engineering. New York: ACM Press, 2006: 361–370.

[18] TAMRAWI A, NGUYEN T T, AL-KOFAHI J M, et al. Fuzzy set and cache-based approach for bug triaging[C]//The 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering. New York: ACM Press, 2011: 365–375.

[19] YANG G, ZHANG T, LEE B. Towards semi-automatic bug triage and severity prediction based on topic model and multi-feature of bug reports[C]//IEEE Computer Software & Applications Conference. Piscataway: IEEE Press, 2014.

[20] XIA X, LO D, DING Y, et al. Improving automated bug triaging with specialized topic model[J]. IEEE Transactions on Software Engineering, 2016, 43(3): 272–297.

[21] LEE S R, HEO M J, LEE C G, et al. Applying deep learning based automatic bug triager to industrial projects[C]//The 2017 11th Joint Meeting on Foundations of Software Engineering. New York: ACM Press, 2017: 926–931.

[22] MANI S, SANKARAN A, ARALIKATTE R. Deeptriage: exploring the effectiveness of deep learning for bug triaging[C]//The ACM India Joint International Conference on Data Science and Management of Data. [S.l.:s.n.], 2019: 171–179.

[23] ALKHAZI B, DISTASI A, ALJEDAANI W, et al. Learning to rank developers for bug report assignment[J]. Applied Soft Computing, 2020, 95: 106667.

[24] ZHANG T, CHEN J, JIANG H, et al. Bug report enrichment with application of automated fixer recommendation[C]//IEEE/ACM International Conference on Program Comprehension. Piscataway: IEEE Press, 2017.

[25] SUN X B, ZHOU C, YANG H, et al. Developer recommendation for software security bugs[J]. Journal of Software, 2018, 29(8): 2294–2305.

[26] YADAV A, SINGH S K, SURI J S. Ranking of software developers based on expertise score for bug triaging[J]. Information and Software Technology, 2019, 112: 1–17.

[27] GUO S, ZHANG X, YANG X, et al. Developer activity motivated bug triaging: via convolutional neural network[J]. Neural Processing Letters, 2020, 51(3): 2589–2606.

[28] BHATTACHARYA P, NEAMTIU I, SHELTON C R. Automated, highly-accurate, bug assignment using machine learning and tossing graphs[J]. Journal of Systems and Software, 2012, 85(10): 2275–2292.

[29] WU H R, LIU H Y, MA Y T. Empirical study on developer factors affecting tossing path length of bug reports[J]. IET Software, 2018, 12(3): 258–270.

[30] XIA X, LO D, SHIHAB E, et al. Automatic, high accuracy prediction of reopened

bugs[J]. Automated Software Engineering, 2015, 22(1): 75–109.
[31] MI Q, KEUNG J, HUO Y, et al. Not all bug reopens are negative: a case study on eclipse bug reports[J]. Information & Software Technology, 2018, 99: 93–97.
[32] BLEI D M, NG A Y, JORDAN M I, et al. Latent dirichlet allocation[J]. Journal of Machine Learning Research, 2012, 3(4–5): 993–1022.
[33] ASUNCION A, WELLING M, SMYTH P, et al. On smoothing and inference for topic models[J]. arXiv preprint, 2020, arXiv: 1205.2662.
[34] DENTON E, GROSS S, FERGUS R. Semi–supervised learning with context–conditional generative adversarial networks[J]. arXiv preprint, 2016, arXiv:1611.06430.
[35] SALIMANS T, GOODFELLOW I, ZAREMBA W, et al. Improved techniques for training gans[C]//Advances in Neural Information Processing Systems. New York: ACM Press, 2016: 2234–2242.
[36] SAHA R K, KHURSHID S, PERRY D E. An empirical study of long lived bugs[C]//2014 Software Evolution Week–IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR–WCRE). Piscataway: IEEE Press, 2014: 144–153.
[37] JOHNSON R, ZHANG T. Supervised and semi–supervised text categorization using LSTM for region embeddings[J]. arXiv preprint, 2016, arXiv:1602.02373.
[38] YANG Z, YANG D, DYER C, et al. Hierarchical attention networks for document classification[C]//The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. [S.l.:s.n.], 2016: 1480–1489.
[39] CHANG C C, LIN C J. LIBSVM: a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3).
[40] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks[C]//Advances in Neural Information Processing Systems. New York: ACM Press, 2014: 3104–3112.
[41] BAHDANAU D, CHO K, BENGIO Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint, 2014, arXiv:1409.0473.

作者简介

席圣渠（1992– ），男，南京大学计算机科学与技术系博士生，主要研究方向为基于深度学习的软件智能开发方法。

徐锋（1975– ），男，博士，南京大学计算机科学与技术系教授、博士生导师，主要研究方向为智能化软件可信技术。

作者简介

陈鑫（1975– ），男，博士，南京大学计算机科学与技术系副教授，主要研究方向为嵌入式系统、软件建模与分析、软件测试与验证。

李宣东（1963– ），男，博士，南京大学计算机科学与技术系教授、博士生导师，软件学院院长，主要研究方向为软件建模与分析、软件测试与验证。

收稿日期： 2020-10-20

通信作者： 席圣渠，xsq@smail.nju.edu.cn

基金项目： 国家重点研发计划基金资助项目（No.2016YFB1000800）；国家自然科学基金资助项目（No.61672274，No.61702252）；江苏省协同创新中心项目

Foundation Items: The National Key Research and Development Program of China (No.2016YFB1000800), The National Natural Science Foundation of China (No.61672274, No.61702252), Collaborative Innovation Center of Novel Software Technology and Industrialization of Jiangsu Province