噪音过滤和深度学习相结合的安全缺陷报告识别

蒋 远 牟辰光 苏小红 王甜甜
(哈尔滨工业大学计算学部 哈尔滨 150001)

摘 要 随着软件规模和复杂性的增加,不可避免会出现各种各样的软件缺陷,其中安全相关的软件缺陷容易被攻击者利用而可能造成重大的经济与生命财产损失.在软件开发维护过程中一般会采用缺陷报告追踪系统以缺陷报告的形式及时地记录和追踪软件所产生的缺陷.自动识别安全缺陷报告可以快速将缺陷报告仓库中和安全相关的缺陷报告识别出来,帮助修复人员及时发现安全缺陷并优先修复.目前常见的安全缺陷报告自动识别方法主要是基于文本挖掘和机器学习相结合的技术,但是由于安全相关缺陷具有特征复杂以及在实际项目中数量较少的特点,使得传统的基于机器学习的识别模型难以提取和安全相关的深层次语义特征,并且模型训练过程受数据集噪音的影响较大,从而导致模型的泛化性能提升出现瓶颈.为了解决该问题,本文提出了一种噪音过滤和深度学习相结合的安全缺陷报告识别框架,该框架首先使用词嵌入技术获取语料库中所有单词的分布式向量表示,然后采用本文提出的基于生成模型的噪音过滤方法 FSDON(Filtering Semantically Deviating Outlier NSBRs)过滤与安全缺陷报告语义相似并且可能是噪音的非安全缺陷报告,最后使用不同的深度神经网络(LSTM、GRU、TextCNN 和 Multi-scale DCNN)构建安全缺陷报告识别模型,完成安全缺陷报告自动识别任务.本文方法在 5 个不同规模的数据集上进行了实验评估,实验结果表明,相比于目前最先进的基于文本挖掘和机器学习相结合的方法,本文方法在 g-measure 指标上平均提升 8.26%,并且在不同规模的数据集上的性能均优于现有最先进的方法.

关键词 安全缺陷报告识别;生成模型;缺陷报告噪音过滤;深度学习
中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2022.01794

Security Bug Report Detection Via Noise Filtering and Deep Learning

JIANG Yuan MU Chen-Guang SU Xiao-Hong WANG Tian-Tian
(*Faculty of Computing, Harbin Institute of Technology, Harbin* 150001)

Abstract With the increase of the scale and complexity of software, it is inevitable that there will be various software bugs. The security-related software bugs are easy to be exploited by malicious users to launch attacks and cause great damage. In software development and maintenance process, the bug report tracking systems such as Bugzilla are usually used to record and track the bugs in the form of bug reports. The identification of the security bug report automatically quickly identifies the security related bug reports in the bug report tracking systems, which could help the developers to work on fast fixing bugs. Recently, many existing methods for security bug report detection have been gaining much attention to tackle such problems by combining text mining and machine learning. However, owing to the small sample size and complex characteristics of security-related bug reports, it is difficult for most previous work based on machine learning methods to capture deep semantic information from textual fields of bug reports. In addition, previous approaches focus on filtering the noise bug reports from datasets using text mining

收稿日期:2021-01-03;在线发布日期:2021-09-16.本课题得到国家自然科学基金项目(61672191)、"十三五"国家重点研发计划(2017YFC0702204)资助.**蒋 远**,博士研究生,中国计算机学会(CCF)学生会员,主要研究领域为软件仓库挖掘、代码漏洞检测和代码表示等.E-mail:jiangyuan@hit.edu.cn.**牟辰光**,硕士,主要研究方向为软件仓库挖掘和安全缺陷报告识别.**苏小红**(通信作者),博士,教授,中国计算机学会(CCF)高级会员,主要研究领域为智能软件工程、软件漏洞检测、程序表示学习、缺陷分派和定位、克隆检测和代码搜索.E-mail:sxh@hit.edu.cn.**王甜甜**,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为软件工程、程序分析和计算机辅助教育.

models without considering the semantic information, which leads to a bottleneck for further improving the prediction performance of the trained model. In order to address the aforementioned problems, in this paper, we develop a novel framework to predict unknown security bug reports by combining semantic-based noise filtering with deep learning techniques. More concretely, it firstly leverages the word embedding technique to get the dense and low-dimensional vector representation of all words in corpus. Secondly, it leverages the proposed Filtering Semantically Deviating Outlier NSBRs (FSDON) method to filter the non-security bug reports (NSBRs) that have higher similarity with security bug reports (SBRs). Finally, it builds predictive models for SBRs detection based on different deep learning networks (LSTM, GRU, TextCNN and Multi-scale DCNN). This method is evaluated on 5 different datasets, and the experimental results show that the *g*-measure performance of this method can be improved by 8.26% on average compared with the state-of-the-art methods. Overall, the proposed method yields the best performance on all the datasets of different scales.

Keywords　security bug report detection; generation model; noise filtering of bug reports; deep learning

1　引　言

软件缺陷(根据 ISO 9000 定义)是指在软件产品中存在,并且未满足与预期或者规定用途有关要求的瑕疵或者问题等[1].随着软件的规模和复杂性日益增大,不可避免地会出现各种各样的软件缺陷.其中,安全相关的缺陷一旦被攻击者利用,将会对软件系统造成巨大安全威胁.例如,Equifax 的服务器遭受黑客攻击,导致多达 1.43 亿美国人的信用信息遭到泄露,WannaCry 勒索软件利用美国国家安全局的"永恒之蓝"(EternalBlue)工具攻击运行 Windows 操作系统的计算机,使英国医疗急诊室瘫痪,拖延了许多患者的及时治疗,造成了巨大的经济与生命财产损失[2].

为了利于收集和管理软件缺陷,越来越多的软件公司比如 Google、Mozilla 已经建立了自己的缺陷报告追踪系统,以及时地记录和追踪软件所产生的缺陷.用户和测试人员通过提交缺陷报告反馈软件使用中可能存在的缺陷信息,软件质量维护人员在接收并检查缺陷报告的内容后将缺陷分派给修复人员进行修复.随着软件规模的增大,软件缺陷也越来越多,其中安全相关的缺陷更容易被攻击者利用而使软件系统面临安全威胁.但是由于缺乏安全相关的领域知识,缺陷报告提交者往往很难准确判断缺陷报告是否与安全相关,如果在提交报告时将安全相关的缺陷标记为非安全相关,那么势必会贻误安全缺陷修复的时机,使系统遭受严重的安全漏洞威胁.采用人工的方式从大量非安全相关的缺陷报告中找出安全缺陷报告,是一项非常耗时耗力的工作.安全缺陷报告的自动识别可以利用缺陷报告仓库中已有的历史缺陷报告训练识别模型,然后使用该模型预测新提交的缺陷报告是否包含安全相关的缺陷,实现安全缺陷的快速识别,从而缩短安全缺陷的修复周期,避免安全漏洞被攻击者利用而对系统造成重大危害和损失.因此,开发自动化的安全缺陷报告识别方法具有重要意义.

目前常见的解决思路是将安全缺陷报告识别任务转化为一个二分类问题进行处理,主要使用文本挖掘和机器学习相结合的方法,即使用文本挖掘技术将缺陷报告转化成向量表示,然后基于缺陷报告的向量表示训练机器学习模型预测新提交的缺陷报告是安全缺陷报告的可能性.例如,Peters 等人[3]提出的 FARSEC 方法,该方法从安全缺陷报告中提取 TF-IDF 值最高的 100 个词作为安全相关的关键词,并利用提取的安全关键词过滤可能是噪音的非安全缺陷报告,然后将过滤后的缺陷报告转化为维度为 100 的特征向量,其中每一维代表该缺陷报告是否包含相应的安全关键词;最后将历史缺陷报告的特征向量作为机器学习算法的输入,训练安全缺陷报告自动识别模型.然而,这种方法存在一定的缺陷.首先,TF-IDF 值较高的词未必就是和安全相关的词,如果提取的安全关键词不准确,那么对非安全缺陷报告的过滤效果就会大打折扣.其次,由于每个

历史缺陷报告可能只包含少数几个和安全相关的关键词,因此根据安全关键词生成的缺陷报告特征向量包含大量的 0 元素,从而导致向量过于稀疏而无法准确刻画缺陷报告所包含的语义信息.针对 FARSEC 方法可能存在的不准确的数据噪音过滤以及向量表示稀疏的问题,我们在前期工作中[4]提出了一种新型的缺陷报告过滤以及表示框架 LTRWES,该方法首先使用排序模型 $BM25F_{ext}$[5]计算每个 NSBR 与所有 SBR 的内容相关性,然后从 NSBR 中过滤掉与 SBR 内容相关度较高的 NSBR,最后利用在大量缺陷报告文本语料库上训练的 word2vec 模型[6-7]将缺陷报告表示为低维连续的实值向量,进而实现更准确的缺陷报告向量表示.由于该方法对数据噪音和缺陷报告表示稀疏的问题做了很大的改进,所以显著地提升了安全缺陷报告识别任务的性能.

由于安全缺陷的语义特征相对复杂并且在真实项目中安全缺陷的实例较少,使得传统基于机器学习的模型难以提取和安全相关的深层次语义特征,从而限制了预测模型在未知数据上的识别效果.而深度学习在特征提取方面有着独特的优势与潜力,将深度学习应用于解决复杂的文本处理任务的研究也已初现成果.考虑到缺陷报告的文本信息使用自然语言描述,所以可以将深度学习应用于安全缺陷报告识别任务.因此,本文研究基于深度学习的安全缺陷报告语义特征提取方法,并将与传统的基于机器学习的识别方法进行比较,研究多种不同的深度学习模型(例如:LSTM、GRU、TextCNN[8]以及 Multi-scale DCNN[9])在不同实际项目下的识别效果.另外,由于不同的词或短语可能被用来表示相同的语义,导致之前基于安全关键词[3]和基于内容相似度的噪音过滤方法[4]可能无法正确给出部分缺陷报告之间的语义相似度,因此本文提出一种新的基于生成模型的噪音缺陷报告过滤方法 FSDON①,该方法利用安全缺陷报告提取安全语义区域,根据每个非安全缺陷报告中的所有单词从安全语义区域产生的概率计算每个非安全缺陷报告的异常值分数,并且根据异常值分数的大小对非安全缺陷报告进行排序,过滤掉和安全缺陷报告相似的并且可能是噪音的非安全缺陷报告.

本文的主要贡献总结如下:

(1)本文提出了一种新的基于生成模型的噪音缺陷报告过滤方法 FSDON,与以往基于关键词或者内容相似度的过滤方法不同,该方法通过考虑缺陷报告的语义信息,利用基于 sHDP(spherical Hierarchical Dirichlet Process)的生成模型识别安全相关的语义区域,并通过计算非安全缺陷报告中的词从安全语义区域产生的概率量化非安全缺陷报告为噪音缺陷报告的可能性(即异常分数),进而过滤异常分数较高的非安全缺陷报告.

(2)本文将噪音过滤方法和深度学习方法相结合,通过提高缺陷报告训练数据集的质量,提升基于深度学习模型的安全缺陷报告识别方法的性能.另外,相比于以往基于深度学习的安全缺陷报告识别方法,本文考虑了更多的深度学习模型,为在实际应用中进行模型选择提供了有用的指导.

(3)本文在公开的不同规模的安全缺陷报告数据集(包括 4 个小规模项目和 1 个大规模项目)上进行了大量的实验,并且从分类和排序两个角度验证了本文方法明显优于目前最先进的基于文本挖掘和机器学习相结合的安全缺陷报告识别方法(FARSEC 和 LTRWES)以及基于深度学习的安全缺陷报告识别方法,尤其在 g-measure 性能上显著优于上述方法.

本文第 1 节介绍安全缺陷报告识别的研究背景及意义;第 2 节介绍相关研究工作;第 3 节详细介绍本文方法以及方法的具体原理和实现细节;第 4 节介绍数据集、评估指标以及实验设计;第 5 节对实验结果进行分析并给出相关的结论;最后第 6 节总结全文,对下一步工作进行展望.

2 相关工作

本节对现有的安全缺陷报告识别工作进行分析总结.

目前应用于安全缺陷报告识别的主流方法是基于文本挖掘和机器学习相结合的方法,即借助于文本挖掘技术生成缺陷报告的特征向量,然后基于特征向量和机器学习算法训练安全缺陷报告自动识别模型.Gegick 等人[10]首次将文本分析技术用于安全缺陷报告识别任务,通过利用词袋模型将缺陷报告表示成一个词项-文档矩阵,并将该矩阵和相应的标签作为输入,训练统计模型,从而实现安全缺陷报告的自动识别.但是该方法受数据噪音以及类别不平衡的影响,导致安全缺陷报告的识别准确率较低.

Behl 等人[11]提出了一种类似的识别方法,该方法同样使用词袋模型表示缺陷报告,但是使用 TF-IDF

① https://github.com/YuanJiangGit/FSDON.git

计算权重，并且使用朴素贝叶斯训练安全缺陷报告识别模型，实验结果表明结合了 TF-IDF 表示的朴素贝叶斯模型的效果更好. Goseva-Popstojanova 等人[12]首次尝试将无监督的机器学习方法应用到安全缺陷报告识别任务并且与有监督的机器学习方法进行比较，对于每种方法都使用了三种特征向量(Binary Bag-of-Words Frequency, Term Frequency, and Term Frequency-Inverse Document Frequency)表示缺陷报告，并且在不同大小的训练集上进行实验以验证所需数据集的大小，结果表明无监督的机器学习模型效果要差于有监督的机器学习模型，而在有监督机器学习中贝叶斯算法在不同项目上的表现最稳定. 另外，考虑到由于安全隐私问题，部分缺陷报告的详细描述信息无法获得，所以 Pereira 等人[13]提出只基于缺陷报告的标题进行安全缺陷报告识别，该方法对比了逻辑回归、朴素贝叶斯和自适应增强算法，在引入不同程度噪音的情况下，测试了三个方法的鲁棒性，实验结果表明当数据集中的噪音比例较少时对所训练模型的性能基本无影响，但是当噪音的比例超过 35%～50%后，模型的性能会受到不同程度的影响. 上述研究都从不同的角度证明了机器学习模型能够自动学习历史缺陷报告数据，实现安全缺陷报告识别任务. 但是模型的识别效果依赖于高质量的数据集，如果数据集中噪音比例太大，模型识别效果就会受到影响.

为了解决现有数据集安全缺陷报告数量较少的问题，Wu 等人[14]提出了基于迭代投票分类的自动数据标记方法，该方法首先选择在 CVE(Common Vulnerabilities and Exposure)中与安全漏洞有关的缺陷报告作为安全缺陷报告，通过小样本标记后，对输入的其他缺陷报告进行投票分类，以实现安全缺陷报告的自动标记和数据扩充，该方法通过数据增强，提高了机器学习模型在安全缺陷报告识别任务上的效果.

除了通过生成数据来扩充数据集的规模外，部分研究人员还通过过滤数据噪音来提高数据集的质量. Peters 等人[3]提出一种安全缺陷报告自动过滤和识别框架 FARSEC，该框架通过使用 TF-IDF 方法自动提取在安全缺陷报告中出现频率最高的 100 个词作为安全关键词，并且利用安全关键词以及 one-hot 技术表示缺陷报告，考虑非安全缺陷报告因可能包含被误标记的安全缺陷报告而影响机器学习模型的训练性能，因此 FARSEC 使用安全关键词计算每个非安全缺陷报告的得分，从而过滤得分较高的非安全缺陷报告. 实验结果表明基于过滤后的数据集训练的安全缺陷报告识别模型，可以更好地泛化到测试实例上，即噪音过滤能够提高模型在未知数据上的识别效果. 我们在前期工作中[4]针对 FARSEC 方法存在的数据噪音过滤不准确以及离散的缺陷报告表示无法编码缺陷的语义信息的问题，提出了一种改进的噪音过滤和表示框架 LTRWES，该框架首先使用 word2vec[6-7]学习缺陷报告语料库中每个词的低维连续向量表示，然后使用信息检索中的排序算法 $BM25F_{ext}$[5]计算每个非安全缺陷报告和所有安全缺陷报告的文本内容相似度，根据相似度分数对所有非安全缺陷报告进行排序，并使用 multiple selector 和 roulette wheel selector[15]过滤掉一定比例的非安全缺陷报告(即与安全缺陷报告内容相似度较高的非安全缺陷报告)，最后将筛选后的非安全缺陷报告和所有的安全缺陷报告一起作为训练数据集，利用多种机器学习算法训练安全缺陷报告识别模型. 由于 LTRWES 在噪音过滤时考虑了缺陷报告的内容相似度和缺陷的部分语义信息，所以显著地提升了安全缺陷报告识别任务的性能.

目前现有的基于深度学习的安全缺陷报告识别方法不多，Palacio 等人[16]提出了 SecureReqNet 方法，该方法从 CVE、GitHub 和 Wikipedia 系统中收集相关的漏洞描述以及和缺陷相关的文档信息组成训练语料库，然后，在该语料库上训练单词向量模型(即 word2vec)获得单词的嵌入向量表示，最后使用卷积神经网络及其变体训练安全缺陷报告识别模型. 实验结果表明该方法在开源和工业项目上取得了令人满意的效果. 随后，Zheng 等人[17]对深度学习方法在安全缺陷报告识别任务上的效果进行了实证研究，使用 TextCNN 和 TextRNN 方法从缺陷报告中提取用于安全缺陷报告识别的语义特征，与 FARSEC 和 LTRWES 方法不同，该方法主要使用 f-measure 指标而非 g-measure 指标来进行模型性能评估，结果表明基于深度学习的安全缺陷报告检测模型的效果优于大多数传统的机器学习分类算法. 然而，对于和安全领域相关的非平衡数据集，g-measure 由于同时考虑了特异性和敏感性，因此相对于 f-measure 指标更适合评估安全缺陷报告识别模型的性能.

与已有的基于深度学习的安全缺陷报告识别方法不同，本文提出一种新的基于生成模型的噪音过滤方法并将其和深度学习模型相结合用于安全缺陷报告识别，该方法能够更有效地降低数据噪音以及

类别不平衡的影响，构建泛化性能更强的安全缺陷报告识别模型.

3 基于深度学习的安全缺陷报告识别方法

3.1 方法的整体框架

基于噪音过滤和深度学习的安全缺陷报告识别方法框架如图 1 所示，共包含如下 4 个步骤：

(1) 文本预处理以及分布式单词表示. 首先对历史缺陷报告数据中的文本信息(摘要和详细描述)进行预处理，合并预处理后的文本组成关于缺陷报告的专业领域语料库；利用从开源项目挖掘的大量处理后的缺陷报告对语料库进行扩充，然后训练 word2vec 模型，并将语料库中的每个单词映射到连续的低维稠密空间，生成单词的低维实值向量表示；

(2) 基于生成模型的噪音过滤. 通过生成模型识别和安全缺陷报告相关的语义区域，计算每个非安全缺陷报告的异常值分数，并且根据异常值分数的大小对非安全缺陷报告进行排序，过滤掉和安全缺陷报告语义相似的并且可能是噪音的非安全缺陷报告；

(3) 基于深度学习的安全缺陷报告识别. 合并过滤后的非安全缺陷报告和所有安全缺陷报告组成训练数据集，并且根据步骤(1)预训练的 word2vec 模型将数据集中的每个缺陷报告转化为特征矩阵，其中每一行代表缺陷报告中每个词的向量表示. 最后，利用深度学习方法(例如：TextCNN、LSTM、GRU、Multi-scale DCNN 等)在过滤后的历史缺陷

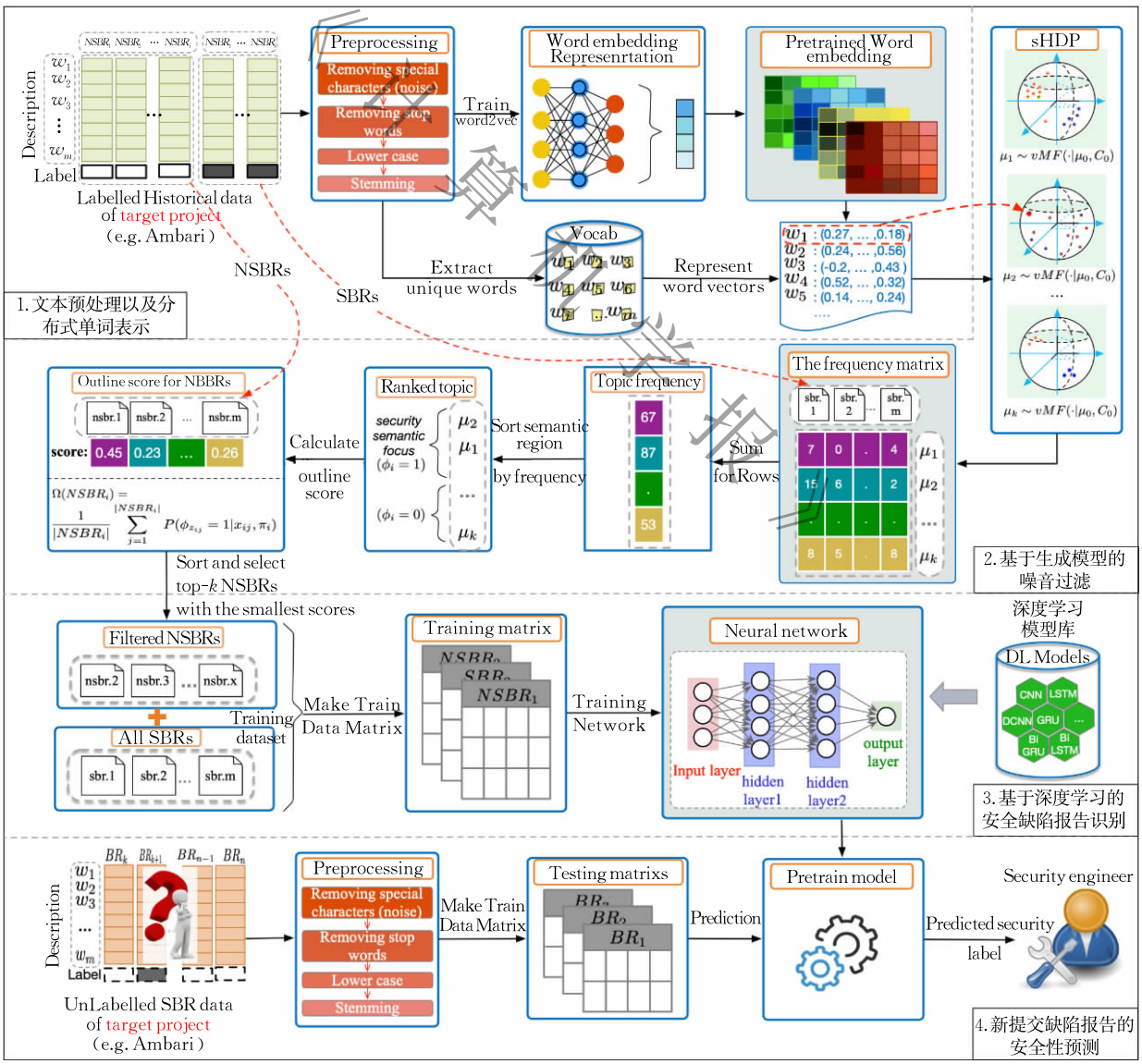

图 1 基于噪音过滤和深度学习的安全缺陷报告识别模型整体框架(框架包含四个阶段：文本预处理和单词分布式向量表示、基于生成模型的噪音过滤、基于深度学习的安全缺陷报告识别以及新提交缺陷报告的安全性预测)

报告向量表示上训练安全缺陷报告识别模型；

(4) 新提交缺陷报告的安全性预测. 对于新提交的缺陷报告，使用和步骤(1)相同的文本处理方法获得缺陷报告预处理后的文本信息，同样采用预训练的 word2vec 模型将文本信息转化为特征矩阵，最后利用已训练的深度学习模型预测该缺陷报告属于安全缺陷报告的可能性.

3.2　文本预处理和分布式单词表示

3.2.1　文本预处理

该阶段主要对历史缺陷报告数据中的文本信息进行预处理，合并预处理后的文本组成关于缺陷报告的专业领域语料库，本文的文本预处理主要包括如下 3 步：

(1) 移除噪音词

缺陷报告的文本描述信息中包含了大量的噪音词，以往研究表明文本中的噪音会阻碍深度学习模型提取缺陷报告的语义特征，因此，在模型训练之前需要对缺陷报告的文本内容进行预处理，主要包括去除 HTML 链接、代码堆栈信息、十六进制数字、文件名以及非字母符号等. 常用的噪音处理方法是使用正则表达式(如表 1 所示)匹配噪音在缺陷报告中的位置，然后将噪音替换成空字符串.

表 1　用于文本信息噪音过滤的正则表达式

表达式	描述
http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+	移除网址信息
Stack trace:	移除栈追踪信息
(\w*)0x\w+	移除 16 进制信息
\.(?!((c|h|cpp|py)\s+$))	移除文件名信息
[^a-zA-Z\.]	移除非字母信息

(2) 移除停用词

停用词是指对提取安全相关特征没有帮助的一些词语，比如“and”、“but”、“I”等，移除停用词可以减少训练 word2vec 模型的开销，同时可以减少安全相关特征提取时的噪音，使得最终学到的缺陷报告的向量表示更加准确. 本文使用 NLTK① 提供的停用词集合对数据集中每个缺陷报告的文本进行处理，去除文本中出现在停用词集合中的词.

(3) 词干提取和小写转化

词干提取是去除缺陷报告中单词的词缀得到词根的过程，比如 follows 和 following，经过词干提取后二者都是 follow，词干提取可以对词的不同形态进行统一归并，本文采用 Snowball 方法②进行词干提取. 小写转化可以将每个单词的字母全部变为小写，对单词的大小写形式进行统一，减少需要训练的单词数量，从而减少模型训练开销.

表 2 是文本预处理前的 Chromium 项目 ID 为 710 的缺陷报告，表 3 是对应表 2 进行文本预处理后的缺陷报告信息.

表 2　预处理前的 Chromium 项目 ID 为 710 的缺陷报告

issue_id	summary	description
710	Issue 710: proxy auto- config file (*.pac) does not seem to work 2 problem?	1. set the proxy auto-config url; 2. access a site that requires that pac file; 3. does not work access fails access should succeed

表 3　预处理后的 Chromium 项目 ID 为 710 的缺陷报告

issue_id	summary	description
710	proxi, auto, config, file, pac, seem, work, probl	set, proxi, auto, config, url, access, site, requir, pac, file, work, access, fail, access, succeed

3.2.2　单词的分布式表示

由于基于机器学习和深度学习的方法将向量(数字数组)作为模型的输入，因此需要将原始缺陷报告的文本内容转化为符合模型输入的特征向量表示. 之前的研究对缺陷报告的向量表示方法有不同的处理，其中，FARSEC 方法使用 one-hot 对缺陷报告进行向量表示，该方法首先获取在安全缺陷报告中出现最频繁的 100 个词作为安全相关关键词，然后利用安全关键词以及 one-hot 技术表示缺陷报告. 由于 FARSEC 使用 TF-IDF 方法自动提取安全相关的关键词，所以存在安全关键词提取不准确的问题[18]. 另外，使用 one-hot 表示的向量具有数据稀疏性，因此本文使用连续的低维稠密向量表示缺陷报告. 目前获取低维向量表示的方法有很多，包括 word2vec[6-7]、fasttext[19] 以及 glove[20] 等，其区别如下所述.

① http://www.nltk.org/

② http://www.nltk.org/_modules/nltk/stem/snowball.html

Word2vec[6-7]是近年来在自然语言处理中常用的技术.它使用浅层神经网络以无监督的方式从大规模文本语料库中学习词汇之间的语义关系.常用的 word2vec 分为两种模型,一种是连续词袋模型(CBOW),该模型根据输入的上下文来预测中心词,另一种是 Skip-gram 模型,其根据输入的中心词去预测上下文.Skip-gram 模型的本质是计算输入单词和输出单词的余弦相似度,然后利用 Softmax 函数进行归一化,但是如果词典中包含大量的词汇,执行上述操作的复杂性较高,会很大程度的影响模型的训练效率,因此,word2vec 引入了两种优化技术即层次 Softmax 和负采样,用于缩短模型的训练时间.

Fasttext[19]是 Facebook 在 2016 年提出的一种针对句子分类的神经网络,它与 CBOW 的结构类似,同样使用文本上下文预测目标出现的概率,不同之处是 fasttext 的输入是更细粒度的字符级 n-gram 信息,并且采用有监督的方法训练网络的参数.由于 fasttext 和 word2vec 都使用了层次 Softmax 降低训练时的复杂度,因此两种方法都具有较高的训练效率.但是对于安全缺陷报告识别任务,由于无法获得缺陷报告文本内容所包含的每条语句的细粒度标签,因此该方法无法用于本文任务去学习单词的向量表示.

Glove[20]通过构建词的共现矩阵学习词向量表示,共现矩阵的构造过程基于全局语料库,因此相比于 word2vec 的向量表示,由共现矩阵分解得到的词向量考虑了更多的全局语义信息.但是 word2vec 训练过程更简单,并且有研究表明 word2vec 训练效果在某些任务上优于 glove.因此,本文采用由 word2vec 训练的词向量作为深度学习模型的输入,并使用反向传播算法调整词向量的参数.

为了训练 word2vec 模型,我们通过挖掘开源项目的缺陷报告追踪系统来收集大量的缺陷报告数据,并结合目标项目的历史缺陷报告组成大规模的缺陷报告文本语料库,利用 3.2.1 节介绍的文本处理技术对语料库中的缺陷报告进行预处理,将清洗后的缺陷报告文本作为数据集训练词嵌入模型,获取单词之间的语义关系,从而将离散的单词符号的语义信息映射到低维稠密实值的向量空间中(即词嵌入空间),使得语义相似的两个词在空间中的分布距离较近.

3.3 基于生成模型的噪音缺陷报告过滤方法

FARSEC 和 LTRWES 等机器学习方法为了提高训练数据集质量,都提出了相应的噪音过滤方法,不同的是 FARSEC 首先使用安全关键词计算每个非安全缺陷报告是噪音缺陷报告的概率,然后过滤掉概率值较大的非安全缺陷报告,以提升数据集的质量.但是该方法由于存在安全关键词提取不够准确的问题,所以可能造成数据噪音过滤也不准确[18].而 LTRWES 使用基于内容相关性进行噪音过滤,该方法借助 learning to rank 技术计算每个非安全缺陷报告和所有安全缺陷报告之间的内容相似度,然后基于计算的内容相似度对非安全缺陷报告进行过滤,筛选掉相似程度高的可能是噪音的非安全缺陷报告.虽然该方法使用内容相似度对噪音进行过滤,改进了以往方法针对安全相关关键词计算不准确的问题,但是仍然存在以下问题:

(1)不同的词或短语可能被用来表示相同的语义,造成基于内容相似度的计算方法可能无法正确给出部分缺陷报告之间的语义相似度;

(2)无法找到合适的词或者短语来描述数据集中安全缺陷的语义内容,造成模型的可解释性差;

(3)缺陷报告中可能含有非常丰富的噪音符号,并且这些符号大部分对于判定该缺陷报告是否是误标记没有任何帮助.

针对上述问题,本文提出一种基于生成模型的噪音缺陷报告过滤方法 FSDON,该方法能够有效地过滤可能是误标记的非安全缺陷报告,其工作流程如下:

(1)首先使用预训练的词嵌入模型将目标项目历史缺陷报告中的所有单词映射为词向量表示,对所有的词向量进行标准化,并使用 von Mises-Fisher(vMF)分布来刻画词嵌入空间中向量的分布.然后,使用基于 sHDP[21]的生成模型识别词嵌入空间中的语义区域;

(2)其次,提取训练数据集中所有安全缺陷报告的文本内容,组成安全缺陷语料库.对安全缺陷语料库中的词所属的语义区域进行统计,得到每个语义区域在所有安全缺陷报告中出现的频率,按照频率的大小对语义区域进行排序,选择前 top-k 个语义区域作为安全相关的语义区域,本文称之为安全语义焦点(Security-related semantic focus);

(3)最后,根据生成的安全语义焦点,为每个非安全缺陷报告计算异常分数,异常分数越高说明非安全缺陷报告包含安全缺陷的可能性越大,即有可能是误标记为非安全缺陷报告的安全缺陷报告.根据异常分数的大小对所有的非安全缺陷报告进行排序,以一定的比率过滤掉异常分数较大的可能是噪音实例的非安全缺陷报告.

3.3.1　生成语义区域

由于每个缺陷报告的文本描述内容在报告提交时都被显式地标记为 Summary 和 Description，因此，我们可以提取训练数据集中所有缺陷报告的文本描述信息并以此组成目标项目语料库. 然后，使用预训练的词嵌入模型将目标项目语料库中的所有单词映射为词向量表示. 另外，由于在自然语言处理中余弦相似度经常被用来衡量两个单词或者短语的语义相似度. 因此为了忽略每个词向量的大小，本文对每个单词向量进行如下的标准化处理，

$$X_{ij}=\boldsymbol{f}(w_{ij})/\|\boldsymbol{f}(w_{ij})\| \tag{1}$$

其中 w_{ij} 表示训练数据集中第 i 个缺陷报告的第 j 个单词，$\boldsymbol{f}(w_{ij})$ 表示通过词嵌入模型输出的向量表示，经过式(1)可以将每个单词表示为与词嵌入向量相同方向的单位向量.

另外，为了刻画不同的语义区域，本文使用 von Mises-Fisher(vMF)分布来表示词嵌入空间中向量的分布，并且使用 vMF 分布的参数来描述每个区域的聚集(Concentration)程度，使用 vMF 分布的原因有以下几点：

首先，vMF 分布的概率分布函数使用了余弦相似度，这和我们使用余弦相似度衡量词汇之间的语义相关性是一致的；其次，经过式(1)的处理，词向量被标准化为分布在球面上的单位向量，这与 vMF 是一个球形分布一致；最后，基于 sHDP 的生成模型依赖于 vMF 分布识别词嵌入空间中可能的语义区域，因此使用 vMF 分布刻画词嵌入空间中的向量表示有利于接下来语义区域的识别操作.

vMF 的概率密度函数如式(2)所示，其中 $I_{v/2-1}(\cdot)$ 是改进的第一种贝塞尔函数[22]，$v/2-1$ 代表贝塞尔函数的阶数. vMF 分布的两个参数是 μ 和 κ，μ 表示分布聚集的平均方向，κ 表示该分布的密集程度，κ 越大表示分布的越密集.

$$\begin{aligned}&p(x)=C_v(\kappa)\exp(\kappa\mu^{\mathrm{T}}x),\\&C_v(\kappa)=\kappa^{v/2-1}(2\pi)^{v/2}I_{v/2-1}(\kappa)\end{aligned} \tag{2}$$

本文采用基于 sHDP 的生成模型来识别缺陷报告语料库中潜在的主题(或者语义区域)信息. 关于该模型的形式化描述如式(3)所示，其中 T 代表语义区域的个数，$|D|$ 代表训练数据集中缺陷报告的数量，$|d_i|$ 表示训练数据集中第 i 个缺陷报告的单词数量. 每个语义区域 μ_t 都是从一个先验 vMF 分布中产生，μ_0 和 C_0 是初始化的参数，每个语义区域的聚集程度 κ 从参数为 m_0，σ_0^2 的先验对数正态分布中产生，π_i 表示第 i 个缺陷报告的语义区域的分布，它服从参数为 α 的狄利克雷分布，即语义区域分布 π_i 由参数为 α 的狄利克雷分布产生，z_{ij} 表示第 i 个缺陷报告第 j 个单词的语义区域类别，它服从参数为 π_i 类别分布，从 π_i 分布中选择一个确定的语义区域，其对应的单词分布是唯一确定的，服从类别分布，$\boldsymbol{x}_{ij}$ 表示第 i 个缺陷报告第 j 个单词的向量表示，其服从参数为 $\mu_{z_{ij}}$ 和 $\kappa_{z_{ij}}$ 的 vMF 分布.

本文使用的基于 sHDP 的生成模型和传统的生成模型(例如主题模型)的最大区别是基于 sHDP 的生成模型使用 vMF 分布生成文档中每个单词的标准化后的分布式向量表示，而传统主题模型则使用多项式分布生成具体的单词本身，仅适用于离散的情况，不适用于本文的应用场景.

$$\begin{aligned}&\mu_t\sim vMF(\cdot\,|\mu_0,C_0),\ t=1,2,\cdots,T,\\&\kappa_t\sim logNormal(\cdot\,|m_0,\sigma_0^2),\ t=1,2,\cdots,T,\\&\pi_i\sim Dirichlet(\cdot\,|\alpha),\ i=1,2,\cdots,|D|,\\&z_{ij}\sim Categorical(\cdot\,|\pi_i),\ j=1,2,\cdots,|d_i|,\\&\boldsymbol{x}_{ij}\sim vMF(\cdot\,|\mu_{z_{ij}},\kappa_{z_{ij}}),\ j=1,2,\cdots,|d_i|\end{aligned} \tag{3}$$

3.3.2　识别安全语义焦点

为了获得和安全相关的语义区域(即安全语义焦点)，本文通过提取训练数据中所有安全缺陷报告的文本内容(即摘要和详细描述)组成目标项目安全缺陷语料库，对安全缺陷语料库中的词所属的语义区域进行统计，得到每个语义区域在所有安全缺陷报告中出现的频率，按照频率的大小对语义区域进行排序，越和安全缺陷相关的语义区域越靠近列表的顶端. 本文选择前 top-k 个语义区域作为安全语义焦点，并使用二元变量 $\phi_t\in\{0,1\}$ 表示第 t 个语义区域是否是一个安全语义焦点，其中 1 代表该区域是安全语义焦点，0 则代表不是. ϕ_t 会被用于后续文档异常分数的计算.

3.3.3　非安全缺陷报告噪音过滤

如果一个非安全缺陷报告中的词汇有很大的比例来源于安全语义焦点，则此非安全缺陷有可能是误标记的缺陷报告，在模型训练之前需要对此类非安全缺陷报告进行过滤，以避免噪音实例对模型的训练过程产生影响. 为了量化非安全缺陷报告是误标记的安全缺陷报告的可能性，即非安全缺陷报告的异常程度，本文提出基于安全语义焦点的非安全缺陷报告异常评估方法. 具体计算过程分为以下两个步骤：首先计算每个词从安全语义焦点中产生的概率，计算第 i 个非安全缺陷报告中单词 j 从安全语义焦点中产生的概率的公式如式(4)所示.

$$P(\phi_{z_{ij}}=1|\boldsymbol{x}_{ij},\pi_i)=\frac{\sum_t \phi_t \pi_i^{(t)} vMF(\boldsymbol{x}_{ij}|\mu_t,\kappa_t)}{\sum_t \pi_i^{(t)} vMF(\boldsymbol{x}_{ij}|\mu_t,\kappa_t)} \quad (4)$$

其中，ϕ_t是一个二元变量，表示该语义区域 t 是否是一个安全语义焦点，$\pi_i^{(t)}$表示第 i 个缺陷报告属于第 t 个语义区域的概率，$vMF(\boldsymbol{x}_{ij}|\mu_t,\kappa_t)$表示从语义区域 t 下产生第 i 个非安全缺陷报告中的第 j 个单词的分布式表示的概率.

然后，根据非安全缺陷报告中每个词由安全语义焦点产生的概率计算该非安全缺陷报告由安全语义焦点产生的概率. 具体计算公式如式(5)所示，其中 $NSBR_i$是训练数据集中的第 i 个非安全缺陷报告，$\boldsymbol{x}_{ij}$是 $NSBR_i$中的第 j 个单词. 使用式(4)计算 $\boldsymbol{x}_{ij}$由安全语义焦点产生的概率 $P(\phi_{z_{ij}})$，重复上述计算，分别求出生成 $NSBR_i$中包含的所有单词的概率，然后求和取平均，作为非安全缺陷报告 $NSBR_i$的异常分数.

$$\Omega(NSBR_i)=\frac{1}{|NSBR_i|}\sum_{j=1}^{|NSBR_i|} P(\phi_{z_{ij}}=1|\boldsymbol{x}_{ij},\pi_i) \quad (5)$$

3.4 基于深度学习的安全缺陷报告识别

本文主要考虑了四种深度学习模型，包括 LSTM、GRU、TextCNN 以及 Multi-scale DCNN 神经网络模型，其中 LSTM 和 GRU 属于递归神经网络的变体，而 TextCNN 和 Multi-scale DCNN 则是基于卷积神经网络的改进.

3.4.1 递归神经网络

由于全连接网络不能获取缺陷报告的时序信息，导致在安全缺陷报告识别任务中的效果受到限制. 为了获取时序信息以提升模型捕获缺陷报告深层语义特征的能力，本文考虑使用递归神经网络(Recurrent Neural Networks，RNN)及其各种变体作为安全缺陷报告语义特征的提取器. 递归神经网络将时序的概念引入到网络的结构设计中，网络中每个时间步对应缺陷报告中的一个单词的输入，并且每个时间步都有一个隐藏状态，当前时刻的隐藏状态只依赖于上一时刻的状态，通过时间步的迭代使得递归神经网络可以提取句子的时序信息. 但是 RNN 只能处理一定间隔的信息，如果缺陷报告的文本内容太长，会出现梯度消失或者梯度爆炸的问题，造成 RNN 在长文中难以训练. 为了解决 RNN 无法学习长距离依赖关系的问题，有研究学者(Hochreiter 和 Schmidhuber)提出一种特殊的递归神经网络 LSTM(Long Lhort-Term Memory)[23]，LSTM 通过改进细胞状态和引入门控制结构以更好地保留序列信息. LSTM 网络的状态更新的形式化描述如下所述：

$$\begin{aligned}
i_t&=\sigma(\boldsymbol{W}_i\cdot[h_{t-1},x_t]+b_i),\\
\tilde{C}_t&=\tanh(\boldsymbol{W}_C\cdot[h_{t-1},x_t]+b_C),\\
f_t&=\sigma(\boldsymbol{W}_f\cdot[h_{t-1},x_t]+b_f),\\
C_t&=f_t\odot C_{t-1}+i_t\odot\tilde{C}_t,\\
o_t&=\sigma(\boldsymbol{W}_o\cdot[h_{t-1},x_t]+b_o)\\
h_t&=o_t\odot\tanh(C_t)
\end{aligned} \quad (6)$$

其中 i_t、f_t和 o_t分别表示在第 t 时间步的输入、遗忘和输出门，C_t和 h_t分别表示在第 t 时间步下的细胞状态和隐藏状态，$\odot$表示逐元素相乘，$\boldsymbol{W}_i$，$\boldsymbol{W}_C$，$\boldsymbol{W}_f$，$\boldsymbol{W}_o$代表权重矩阵，b_i，b_C，b_f，b_o代表偏移项，σ和 tanh 是激活函数. 最后，通过迭代计算所有时间步的隐藏状态，获取每个句子的时序信息.

基于 LSTM 的安全缺陷报告识别模型的总体框架如图 2 所示. 所述模型的输入是经过文本预处

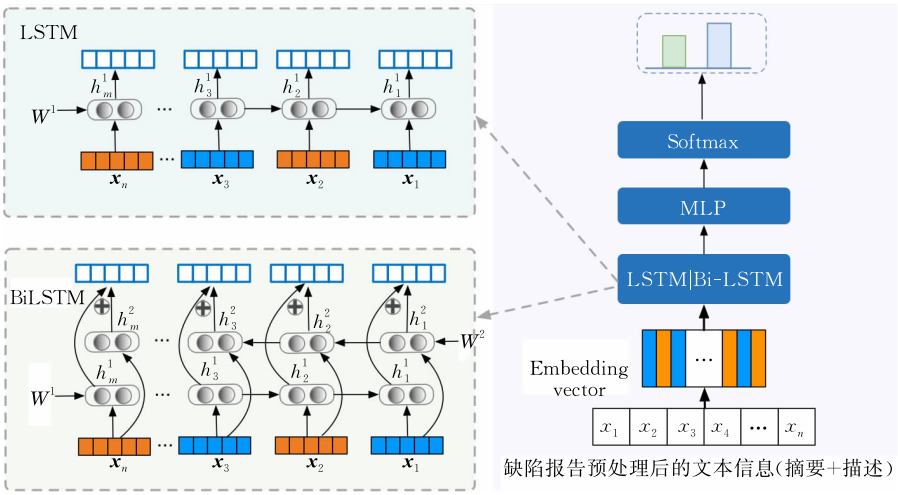

图 2 基于 LSTM 和 BiLSTM 的安全缺陷报告识别模型

理后的缺陷报告的摘要和描述信息(由 n 个单词组成,表示为 $x_{1:n}=[x_1;x_2;\cdots;x_n]$),然后经过词嵌入层将 $x_{1:n}$ 转化为分布式的向量表示(本文用粗体表示相应符号的向量,例如 $\boldsymbol{x}_1$ 是单词 x_1 的向量表示),将单词嵌入向量 $\boldsymbol{x}_{1:n}$ 按照时间步的顺序传入 LSTM 网络,根据式(6)调整 n 个时间步的隐藏状态,然后对 n 个隐藏状态进行最大池化(Max-Pooling)以有效提取缺陷报告中最具代表性的特征信息并输出缺陷报告的向量表示,最后将缺陷报告向量表示作为全连接层的输入,并且利用 Softmax 分类计算该缺陷报告属于安全缺陷报告的概率. 输出概率较大的标签作为模型预测的结果,例如当输出标签为 0 时,代表该缺陷报告是一个非安全缺陷报告;当输出标签为 1 时,则表明该缺陷报告是安全缺陷报告.

为了更好地捕获缺陷报告的上下文信息,本文也考虑采用 BiLSTM 来提取缺陷报告更丰富的语义特征. 相比于 LSTM,BiLSTM 在时间步 t 的隐藏状态(h_t)是由前向网络的隐藏状态($\overrightarrow{h}_t$)和后向网络的隐藏状态($\overleftarrow{h}_t$)拼接而成,具体公式如式(7)所示:

$$\begin{aligned}&\overrightarrow{h}=\overrightarrow{LSTM}(x_t),\ t\in[1,n],\\&\overleftarrow{h}_t=\overleftarrow{LSTM},\ t\in[n,1],\\&h_t=[\overrightarrow{h}_t,\overleftarrow{h}_t],\ t\in[1,n]\end{aligned}\tag{7}$$

除了 LSTM 之外,另一种类型的循环神经网络 GRU(Gate Recurrent Unit)[24] 也被广泛应用于自然语言处理任务中,相比于 LSTM,GRU 网络没有使用细胞状态,仅使用隐藏状态来记忆输入数据的语义信息,因此 GRU 参数更少,模型更加简单,在某些复杂文本处理任务上效果优于 LSTM. 为了验证在安全缺陷报告识别任务上,GRU 是否明显优于 LSTM,本文同样考虑选择 GRU 作为候选的安全缺陷报告识别模型,并在标准数据集上进行广泛的实验研究.

3.4.2　卷积神经网络

在自然语言处理领域,卷积神经网络(Convolution Neural Network,CNN)[25] 已经被成功应用于文本分类任务,并且取得了良好的效果. 其中最著名的模型是 Kim 等人在 2014 年提出了用于文本分类的卷积神经网络 TextCNN[8],本文将其应用于安全缺陷报告识别任务,基于 TextCNN 的安全缺陷报告识别模型的总体框架如图 3 所示.

首先,模型的输入为包含有 n 个单词的缺陷报告,表示为 $x_{1:n}=[x_1;x_2;\cdots;x_n]$,然后经过词嵌入层将 $x_{1:n}$ 转化为分布式的向量表示,不同于将 CNN 应用于图片识别,由于词向量不可拆分,因此在 TextCNN 中卷积核的宽度就是词向量的维度,卷积核长度代表每次卷积的单词个数,框架图中给出的卷积核长度是 2 和 3,即分别使用 m 个两种长度卷积核进行卷积,对卷积层输出的特征向量进行最大池化以有效提取缺陷报告中最具代性的特征信息,对最大池化后的值进行拼接得到缺陷报告的向量表示. 将缺陷报告的向量表示传入全连接层,然后利用 Softmax 分类计算该缺陷报告属于安全缺陷报告的概率,并取概率较大的标签作为网络的输出.

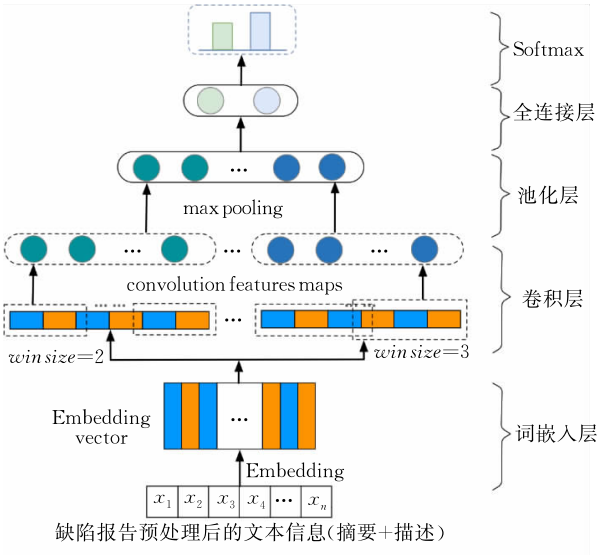

图 3　基于 TextCNN 的安全缺陷报告识别模型

卷积神经网络虽然能有效地提取特征,但随着层数的增多,模型变得越来越复杂,训练效率也由于梯度消失问题变的越来越低. 为了解决该问题,本文提出利用多尺度特征注意力的密集连接卷积神经网络(Multi-scale DCNN)来提取缺陷报告更深层次的语义特征. 网络的整体架构如图 4 所示.

如图 4 所示,Multi-scale DCNN 由三部分组成,分别是 DenseBlock、多尺度注意力机制和语义整合层(LSTM). 其中 DenseBlock 用于获取缺陷报告的 n-gram 语法信息,例如第一层网络用于获取缺陷报告的 1-gram 语法,第二层则用于获取 2-gram 语法. 多尺度注意力机制能够自动选择不同尺度的多元语法特征,即通过对单词在不同层的向量表示进行加权求和,实现精确捕捉单词上下文丰富多元语法特征的功能. 语义整合层利用长短时记忆网络整合所有单词的多尺度特征生成包含序列依赖关系的缺陷报告全局特征表示.

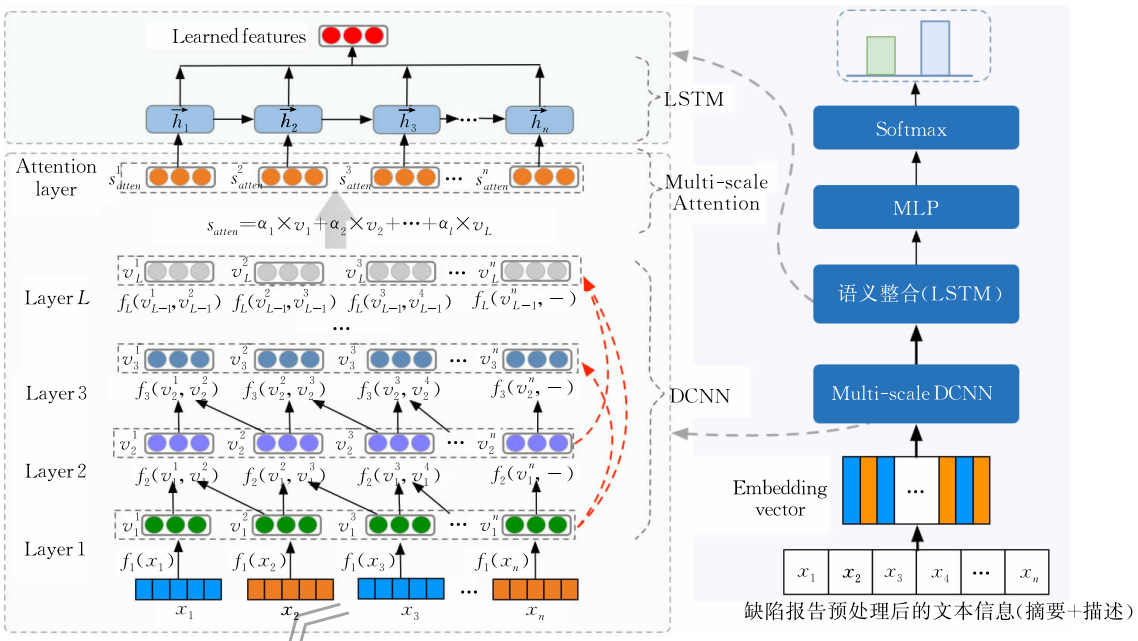

图 4 基于 Multi-scale DCNN 的安全缺陷报告识别模型

首先,将预处理后的缺陷报告文本信息 $x_{1:n}=[x_1;x_2;\cdots;x_n]$ 传入词嵌入层,得到每个单词的分布式向量表示 $\boldsymbol{x}_{1:n}$,然后通过窗口大小为 1 的卷积层(Layer1)获取缺陷报告的 1-gram 信息(即 unigram 信息),之后的卷积层(即 Layer2~L)采用窗口大小为 2 的卷积核提取缺陷报告多元语法特征,如图 4 所示,Layer2 用于提取缺陷报告的 2-gram(即 bigram)语法,Layer3 用于提取缺陷报告的 3-gram(即 trigram)语法,其中 f 是由三个级联(包含 conv、batch normalization 以及 relu)操作组成的组合函数用于提取底层的语法(gram)信息生成更高层的语法特征. 例如,基于 Layer2 生成的 2-gram 信息 $f_2(x_1,x_2)$ 和 $f_2(x_2,x_3)$ 可以组合成 3-gram 信息 $f_3(x_1,x_2,x_3)$.

通过 DenseBlock 可以获得不同大小尺度(即 1-gram~L-gram)的语法特征,本文使用多尺度注意力机制实现不同尺度的多元语法特征的自动选择,即通过对单词在不同层的向量表示进行加权求和,实现精确捕捉单词上下文丰富多元语法特征的功能,具体计算公式如式(8)所示.

$$\begin{aligned} s_l^i &= F_{ensem}(\boldsymbol{v}_l^i)=\sum_{j=1}^{k}\boldsymbol{v}_l^i(j),\\ s^i &= [s_1^i,s_2^i,\cdots,s_L^i],\\ \alpha^i &= \mathrm{Softmax}(MLP(s^i)),\\ \boldsymbol{v}_{attn}^i &= \sum_{l=1}^{L}\alpha_l^i\boldsymbol{v}_l^i \end{aligned} \tag{8}$$

其中,$\boldsymbol{v}_l^i$ 表示第 l 层输出的关于第 i 个单词的特征向量,由于特征向量的每个元素都是由固定窗口大小的卷积核提取的语义特征值,因此可以将 v_l^i 的所有维度的元素求和得到特征向量的描述符 s_l^i. 当得到第 i 个单词在所有层的特征描述符($[s_l^i,s_2^i,\cdots,s_L^i]$)之后,本文使用全连接网络预测第 i 个单词针对 L 个不同尺度的多元语法特征的权重,使用权重对单词 i 在不同层的向量表示进行加权求和得到单词 i 的最终特征向量表示 $\boldsymbol{v}_{attn}^i$.

本文将缺陷报告中所有单词的多尺度注意力特征向量 $\boldsymbol{v}_{attn}^i$ 传入长短时记忆网络中,获得该缺陷报告整体的特征表示,最后将缺陷报告的特征向量传入全连接层,然后利用 Softmax 分类计算该缺陷报告属于安全缺陷报告的概率,取概率较大的标签作为模型的输出.

4 实验设计

为了验证本文提出的基于生成模型的噪音过滤和深度学习模型相结合的安全缺陷报告识别方法的有效性,我们设计了如下 4 个研究问题(Research Question,RQ):

RQ1. 基于生成模型的噪音过滤和深度学习相结合的安全缺陷报告识别方法是否优于以往基于深度学习的安全缺陷报告识别方法?

RQ2. 基于生成模型的噪音过滤和深度学习相结合的安全缺陷报告识别方法是否优于传统的基

于文本挖掘和机器学习相结合的安全缺陷报告识别方法？

RQ3. 本文提出的噪音过滤方法相比于目前最先进的缺陷报告噪音过滤方法是否能更有效地提高基于深度学习的安全缺陷报告识别模型的性能？

RQ4. 本文提出的方法是否能生成更有效的可疑安全缺陷报告列表？

4.1　数据集

本文使用的安全缺陷报告标准数据集从 5 个开源项目中收集，包括 4 个小规模数据集和 1 个大规模数据集. 其中的小规模数据集 Ambari、Camel、Derby 和 Wicket 是由 Ohira 等人[26]提供，对应于 Apache 的四个不同应用领域的开源项目，每个项目均使用 JIRA 作为缺陷报告追踪系统，为了保证该数据集的质量，作者采用人工的方式分别对每个项目中的实例所属的安全标签进行标注. Chromium 数据集是出自 FARSEC 文献[3]，由于 Chromium 数据集中的每个缺陷报告在被用户或者开发人员提交时就被显示地标记为是否和安全相关，因此，无需花费额外的人力对数据实例进行再次标注.

每个数据集均以 CSV 文件形式存储，其中每一行代表一个缺陷报告实例，每一列代表缺陷报告的不同属性信息，包括摘要（summary）、描述（description）、安全（Security）标签等[26]. 根据 Security 标签的值，我们可以把所有的缺陷报告实例分为安全缺陷报告和非安全缺陷报告，当一个缺陷报告的安全标签为 1 时，该缺陷报告是安全缺陷报告，否则该缺陷报告为非安全缺陷报告.

关于标准安全缺陷报告数据集的统计信息如表 4 所示，其中包括每个项目包含的所有缺陷报告的提交时间范围（时间段）、所有缺陷报告的数量、每个项目所包含的安全缺陷报告的数量和安全缺陷报告在所有缺陷报告中所占的比例等. 由表 4 可知，不管是小规模数据集还是大规模数据集，安全缺陷报告所占比重都很小，仅有 0.5%～9%，存在着严重的类别不平衡问题.

表 4　数据集统计信息

数据集	时间段	缺陷报告数量	安全缺陷报告数量	安全缺陷报告比例/%
Ambari	09/26/11～08/08/14	1000	29	3
Camel	07/08/07～09/18/13	1000	32	3
Chromium	08/30/08～06/11/10	41940	192	0.5
Derby	09/28/04～09/17/14	1000	88	9
Wicket	10/20/06～11/09/14	1000	10	1

4.2　评估指标

本文实验所用的评估指标计算方法如表 5 所示，其中 TP 表示真阳性，指测试集中所有的安全缺陷报告被正确预测为安全缺陷报告的数量；FP 表示假阳性，指测试集中将非安全缺陷报告预测为安全缺陷报告的数量；TN 表示真阴性，指测试集中将非安全缺陷报告预测为非安全缺陷报告的数量；FN 表示假阴性，指在测试集中将安全缺陷报告预测为非安全缺陷报告的数量. 基于 TP、FP、TN、FN 的其他指标的详细描述如下所示：

（1）召回率（*recall*）. 指正确识别的安全缺陷报告数量占所有安全缺陷报告的比例，衡量模型对安全缺陷报告的识别能力，召回率越高代表模型识别安全缺陷报告能力越好；

（2）误报率（pf）. 将非安全缺陷报告识别为安全缺陷报告的数量占所有非安全缺陷报告的比例，衡量模型对非安全缺陷报告的误报率，误报率越高代表模型对非安全缺陷报告的识别能力越差. 与误报率相关的一个指标叫特异度（$100-pf$），表示非安全缺陷报告识别正确的数量占所有非安全缺陷报告的比例；

（3）精准率（*prec*）. 识别为安全缺陷报告的样本中实际的安全缺陷报告所占比例；

（4）f-measure. 召回率和精准率的调和平均值，由于二者相互牵制，该指标可以平衡二者，从而进行综合评估；

（5）g-measure. 召回率和特异度（1－误报率）的调和平均值，综合考虑了安全缺陷报告和非安全缺陷报告的召回率；

（6）平均精准率（AP_n）. 在排序列表中前 top-k 个元素精准率的平均值，计算方法如表 5 所示，其中 $P(k)$ 是在排名列表中位置 k 处的精准率，n 是预测为安全缺陷报告的个数；

（7）MAP_n. AP_n 的平均值，用于衡量安全缺陷报告识别模型预测的安全缺陷报告排序列表的性能，如果值越大，说明真实的安全缺陷报告接近排序列表顶端的可能性越大.

表 5 评估指标

		实际	
		SBR	NSBR
预测	SBR	TP	FP
	NSBR	FN	TN
召回率(pd)		$\frac{TP}{TP+FN}$	
误报率(pf)		$\frac{FP}{FP+TN}$	
精准率($prec$)		$\frac{TP}{TP+FP}$	
f-measure		$\frac{2\times pd\times prec}{pd+prec}$	
g-measure		$\frac{2\times pd\times(100-pf)}{pd+(100-pf)}$	
平均精准率(AP_n)		$\sum_{k=1}^{n}\frac{P(k)}{n}$	
MAP_n		$\sum_{i=1}^{N}\frac{AP_{n_i}}{N}$	

自动识别安全缺陷报告的目的是尽可能多地找到有可能包含安全问题的缺陷报告,由于缺陷报告存在严重的类别不平衡问题,即使安全缺陷报告预测准确,但是精准率受非安全缺陷报告的影响较大,因此综合考虑召回率和精准率的 f-measure 无法正确衡量安全缺陷报告识别的性能,而 g-measure 综合了安全缺陷报告和非安全缺陷报告的召回率,即敏感度和特异度,更适合评估在类别不平衡的数据集上训练的安全缺陷报告识别模型,因此本文重点使用 g-measure 指标对模型的性能进行评估. 另外,使用 g-measure 指标方便与目前最先进的方法(例如 FARSEC 和 LTRWES)进行比较. 除了 g-measure 之外,在安全相关的研究中,召回率指标[4,27]已经被证明相比于其他指标更为重要,这是因为安全缺陷报告被正确识别的重要性要高于非安全缺陷报告被正确识别的重要性,即如果安全缺陷报告一旦被错误地识别为非安全缺陷报告,将会给软件系统造成巨大的安全隐患. 因此本文也重点考虑召回率作为模型性能评估和分析的指标.

4.3 实验设置

本文实验中的参数分为数据集参数、噪音过滤参数、模型参数以及模型训练参数,其设置如下所述.

通常在开始构建模型之前需要把数据集进行划分,为了与之前最先进的方法(即 FARSEC 和 LTRWES)的实验设置保持一致,本文将各个项目数据集按照时间排序后,按照 1∶1 将其划分为训练集和测试集. 由于一般缺陷报告的长度较短并且缺陷报告详细描述(Description)的末尾单词包含的信息量相对较少,因此本文选择对数据集中的每个缺陷报告,设置其文本信息的最大长度为 64,即输入模型的缺陷报告的单词数量为 64. 对于含有较长单词序列的缺陷报告,则截断(truncating)其超出的部分,对于较短的单词序列则以 0 填充(padding),使输入模型的文本信息的长度保持一致.

对噪音过滤而言,需要设定过滤的非安全缺陷报告数量,根据以往噪音过滤经验,本文设置需要过滤的非安全缺陷报告的数量为全部非安全缺陷报告数量的 10%,即只过滤 10%的非安全缺陷报告,不同于我们在前期工作中提出的基于内容过滤和机器学习的安全缺陷报告识别方法,本文设置固定的缺陷报告过滤比率,是因为深度学习模型的参数调整过程更加耗时并且会消耗大量的计算资源,每次改变噪音过滤比率都会得到新的数据集,从而需要重新调整深度学习模型的所有超参数,因此考虑实际应用的代价,本文根据以往经验对不同规模的项目设置固定的缺陷报告过滤比率. 另外,由于安全缺陷报告在数据集中占的比例较少,设置和安全相关的语义焦点的个数为 3,设置 Dirichlet 先验折棍(Stick breaking)概率分布的参数 α 为 1. 对深度学习模型来说,网络模型的超参数决定了模型的复杂度和学习能力,对模型的性能产生很大的影响,选择合适的超参数能够让损失函数收敛更快,更易得到泛化性能较高的预测模型. 因此,调整网络的超参数是使深度学习模型实现这一目标的重要步骤之一. 本文采用网格搜索的策略,首先通过设置超参数的搜索空间,然后对每一组可能的超参数组合进行模型的训练和验证,最后通过性能评估得到最优的超参数配置.

本文设置的超参数搜索空间如表 6 所示,需要值得注意的是,对于 Multi-scale DCNN 网络除了设置 DenseBlock 中 CNN 的层数 L 和每层输出的特征向量的维度外,还需要设置与 DenseBlock 相连的语义整合层即 LSTM 网络的参数. 本文使用交叉熵损失和优化器(Adam 或者随机梯度下降(SGD))来调整网络的参数. 相较于 Adam,SGD 训练速度较慢,但是在 Wicket 数据集上训练效果更好. 另外,为了减轻类别不平衡的问题,在利用反向传播算法调整模型参数时,本文使用[150∶1]的代价矩阵计算交叉熵函数的损失值,即误识别的安全缺陷报告的损失和误识别的非安全缺陷报告的损失比是 150∶1,通过给误识别的安全缺陷报告更大的损失权重,目的是使模型更注重和安全相关的缺陷报告识别的准确率.

表 6　模型超参数的搜索空间

超参数(Hyper-parameter)	搜索范围(Value)	描述(Description)
Embedding size (Word2vec)	[60,120,150,200]	Word2vec 模型输出的单词向量维度
RNN hidden size (LSTM,BiLSTM,GRU,Bi-GRU,DCNN)	[64,75,100,125,150,200,250]	循环神经网络隐藏层大小
RNN layer(LSTM,BiLSTM,GRU,Bi-GRU)	[1,2,3,4]	循环神经网络隐藏层的层数
Widths of convolutional filters(TextCNN)	[(1,3,5),(3,5,7),(1,3,5,7),(3,5,7,9),(1,3,5,7,9)],	卷积神经网络可选的窗口组合
The number of Filters(TextCNN)	[60,120,150,200]	卷积神经网络卷积核个数
CNN layer(DenseBlock)	[2,3,4,5,6,7]	DenseBlock 中 CNN 的层数 L
Feature vector size(DenseBlock)	[80,100,120]	DenseBlock 每层输出的特征向量的维度
Dropout	[0.25,0.3,0.4,0.45]	Dropout 参数范围
Initial learning rate	[0.01,0.001,0.0001]	学习率范围
Optimizer	[Adam,SGD]	优化器使用 SGD 和 Adam
Batches of size	[16,32,64,128]	网络训练时的批处理大小
Epoch	[10,20,30,40,50]	网络的训练轮数

5　结果分析

5.1　针对 RQ1 的结果分析

RQ1. 基于生成模型的噪音过滤和深度学习相结合的安全缺陷报告识别方法是否优于以往基于深度学习的安全缺陷报告识别方法?

本节将以往基于深度学习的安全缺陷报告识别方法作为基准方法,在 5 个规模不同的安全缺陷报告标准数据集上进行实验和测试,并将实验结果作为本节实验的对照组. 由于以往的基于深度学习的方法只考虑了 TextCNN 和 LSTM 作为缺陷报告的特征提取器,无法全面地比较不同模型之间的性能差异,因此本文在上述模型的基础上又加入了其他的网络模型或其变体(Bi-LSTM、BRU、Bi-GRU 和 Multi-scale DCNN)作为基准模型,并且对于每种模型,我们均使用在 4.3 节介绍的网格搜索选择最优的超参数组合.

另外,为了对比噪音过滤方法是否能有效地提升深度学习模型在不同项目上的预测性能,我们首先使用本文提出的基于生成模型的噪音过滤方法 FSDON 在模型训练之前对上述项目的训练数据集进行噪音过滤,即通过移除可能是噪音的非安全缺陷报告提升训练数据集的质量,噪音过滤前后的训练数据集的统计信息如表 7 所示. 然后,基于过滤噪音后的训练数据集充分训练上述 6 种深度学习模型. 最后,在项目原始的测试集上评估所训练的模型的性能. 两组实验的对比结果如表 8 所示,其中"—"代表在原始标准数据集上的实验结果,"FSDON"代表模型在经过 FSDON 方法过滤后的标准数据集上的实验结果,加粗字体为每个模型在不同数据集上能够达到的最好 g-measure 值. 此外,为了更详细地了解在每个数据集上被深度学习模型正确预测的实例个数,表 9 给出了在各数据集上取得最优 g-measure 的深度学习模型的评估指标详情,表中以百分比形式显示各个指标值. 由表 9 可知,相比于使用未过滤噪音的标准数据集,使用经过噪音过滤后产生的训练数据集训练的深度学习安全缺陷报告识别模型在 5 个项目的测试数据集上的泛化性能均有提升,从而验证了本文提出的过滤方法在不同项目上的有效性. 以 Ambari 项目为例,在噪音过滤后的数据集上训练的模型比在原始数据集上训练的模型的平均性能(g-measure)提高了 1.81%.

表 7　噪音过滤前后的训练数据集信息

数据集	过滤前		过滤后	
	安全缺陷报告数量	非安全缺陷报告数量	安全缺陷报告数量	非安全缺陷报告数量
Ambari	22	478	22	431
Camel	14	486	14	438
Chromium	76	20893	76	18804
Derby	46	454	46	409
Wicket	4	496	4	447

表 8 不同的深度学习模型在噪音过滤前后的标准数据集上的 *g*-measure 结果

数据集\方法	Filter	BiLSTM	LSTM	Multi-scale DCNN	TextCNN	GRU	BiGRU	Average
Ambari	—	96.10	90.99	92.83	87.74	90.63	90.81	91.52
	FSDON	**97.61**	**93.98**	**96.97**	**89.44**	**90.90**	**91.08**	**93.33**
Camel	—	62.56	60.79	70.39	59.26	65.04	67.47	64.25
	FSDON	**68.15**	**69.96**	**73.10**	**61.26**	**66.54**	**70.31**	**68.22**
Chromium	—	88.67	86.98	87.87	87.15	89.56	88.19	88.07
	FSDON	**88.91**	**87.19**	**89.35**	**88.10**	**89.69**	**88.87**	**88.69**
Derby	—	80.92	81.30	83.62	76.30	82.62	83.14	81.32
	FSDON	**82.05**	**83.62**	**85.10**	**78.91**	**83.41**	**81.83**	**82.49**
Wicket	—	**69.45**	59.51	85.05	**67.24**	50.30	54.68	56.49
	FSDON	69.07	**64.29**	**86.11**	62.40	**66.94**	**63.13**	**68.66**

表 9 在噪音过滤前后的标准数据集上 *g*-measure 效果最优的模型的各个评估指标详情（其中加粗结果代表具有最大的 *g*-measure）

数据集	Filter	*TN*	*TP*	*FN*	*FP*	*pd*	*pf*	*prec*	*f*-measure	*g*-measure
Ambari	—	456	7	0	37	100.00	7.51	15.91	27.45	96.10
	FSDON	**470**	**7**	**0**	**23**	**100.00**	**4.67**	**23.33**	**37.48**	**97.61**
Camel	—	400	11	7	82	61.11	17.01	11.83	19.82	70.39
	FSDON	**390**	**12**	**6**	**92**	**66.67**	**19.09**	**11.54**	**19.67**	**73.10**
Chromium	—	19464	99	16	1391	86.09	6.67	6.64	12.34	89.56
	FSDON	**18729**	**103**	**12**	**2126**	**89.57**	**10.19**	**4.62**	**8.79**	**89.69**
Derby	—	396	34	8	62	80.95	13.54	35.42	49.28	83.62
	FSDON	**387**	**36**	**6**	**71**	**85.71**	**15.50**	**33.64**	**48.32**	**85.10**
Wicket	—	439	5	1	65	83.33	13.16	7.14	13.16	85.05
	FSDON	**440**	**5**	**1**	**51**	**83.33**	**10.93**	**8.47**	**15.38**	**86.11**

除此之外，从表 8 中我们发现对于大规模数据集 Chromium，即便存在噪音和类别不平衡问题，基于深度学习的安全缺陷报告识别方法也可以取得很高的 *g*-measure，例如，不同深度学习模型在 Chromium 数据集上的平均 *g*-measure 可达到 88.07%，其中 GRU 模型取得的效果最好，*g*-measure 值为 89.56%；对于过滤前的小规模标准数据集，不同的深度学习模型的识别效果差异较大，比如在 Ambari 上所有深度学习模型的平均 *g*-measure 值为 91.52%，然而在 Wicket 上只能达到 56.49%. 这是因为 Wicket 数据集不仅可能包含大量的噪音而且含有很少的安全缺陷报告实例（由表 4 可知，Wicket 训练集中只包含 4 个安全缺陷报告，正是由于这种严重的数据不平衡以及潜在的数据噪音的影响，使得深度学习模型不能充分提取训练数据集中和安全缺陷相关的深层语义特征，从而导致模型在未知数据（即测试集）上的泛化能力较差. 另外，即使对于包含相似比例安全缺陷报告的不同项目而言，同一深度学习模型的性能差异也可能较大，例如 BiLSTM 模型在过滤后的 Ambari 数据集上的 *g*-measure 高达 97.61%，而在过滤后的 Camel 只有 68.15%，这是由于不同的数据集本身包含的特征复杂度不同，例如在 Ambari 项目中记录的软件安全缺陷特征相对简单，容易提取，因此深度学习模型能够有效地从缺陷报告的文本字段中学习和安全相关的漏洞语义信息，然而，在 Camel 项目中记录的软件安全缺陷特征可能相对比较复杂，在数据集规模较小的情况下，深度学习模型不易提取深层次的安全语义特征.

此外，为了分析本文提出的噪音过滤方法是否能够显著提高基于不同的深度学习的安全缺陷报告识别模型的性能，本文对表 8 中的实验结果进行了 Wilcoxon 符号秩检验（Wilcoxon Signed-Rank Test），并设置显著性水平 α 为 0.05. 由于计算的 p-value 是 0.0001 并远远小于 0.05. 这表明，基于显著性分析，FSDON 过滤方法能够显著提高多种深度学习模型在安全缺陷报告识别任务上的泛化性能，从而验证了本文提出的噪音过滤方法针对不同深度学习模型的有效性.

5.2 针对 RQ2 的结果分析

RQ2. 基于生成模型的噪音过滤和深度学习相结合的安全缺陷报告识别方法是否优于传统基于文本挖掘和机器学习相结合的安全缺陷报告识别方法？

由于安全缺陷具有特征复杂以及在实际项目中数量较小的特点，使得传统的机器学习方法难以提取和安全相关的深层语义特征，从而导致识别模型的性能提升出现瓶颈. 为了验证本文提出的噪音过

滤和深度学习相结合的安全缺陷报告识别方法是否能更有效地提取缺陷报告的深层语义特征，本节选取目前最先进的基于文本挖掘和机器学习相结合的识别方法 FARSEC 方法和 LTRWES 方法作为基准方法，使用其在各项目数据集上的最优结果与本文提出的噪音过滤和深度学习相结合的方法的识别结果进行比较. 本文使用 *g*-measure 作为主要评估指标，实验结果如表 10 所示，其中机器学习模型 RF、LR、MLP 和 SVM 分别代表随机森林、线性回归、多层感知机和支持向量机. 虽然，已有相关的实证研究[17]证明了深度学习方法要优于基于文本挖掘和机器学习相结合的方法，但是该研究仅和传统方法 FARSEC 进行比较，然而目前性能最好的基于文本挖掘和机器学习相结合的安全缺陷报告识别方法是 LTRWES，并且 LTRWES 实验结果表明该方法明显优于 FARSEC 方法（例如在不同项目上的 *g*-measure 指标平均提升超过 16%）. 因此，只在文献[17]的结论基础上，仍然无法确定基于深度学习的方法是否明显优于目前最先进的基于文本挖掘和机器学习相结合的方法.

表 10　基于文本挖掘和机器学习的 FARSEC、LTRWES 方法和基于生成模型的噪音过滤和深度学习的方法（本文方法）的最优 *g*-measure 对比

数据集	FARSEC		LTRWES		噪音过滤 FSDON 和深度学习	
	机器学习模型	*g*-measure	机器学习模型	*g*-measure	深度学习模型	*g*-measure
Ambari	RF	71.90	LR	86.16	BiLSTM	97.61
Camel	LR	53.80	SVM	65.47	Multi-scale DCNN	73.10
Chromium	MLP	65.40	MLP	86.67	GRU	89.69
Derby	RF	61.70	MLP	75.92	Multi-scale DCNN	85.10
Wicket	LR	65.00	MLP	70.93	Multi-scale DCNN	86.11

另外，为了体现本文的方法（FSDON＋Deep Learning）是否优于目前最先进的安全缺陷报告识别方法 FARSEC 和 LTRWES，我们以柱状图的形式更直观地描绘三种方法在不同项目数据集上的性能差异，实验结果如图 5 所示.

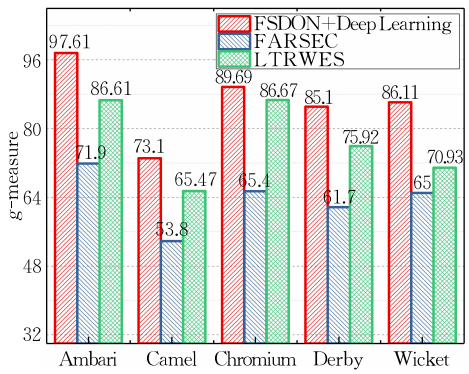

图 5　本文方法与基于文本挖掘和机器学习的方法在各个数据集上 *g*-measure 对比

从实验结果可知，本文提出的噪音过滤和深度学习相结合的方法在 5 个数据集上的 *g*-measure 指标均优于其他两种方法，并且相比 FARSEC 方法，本文的方法在 5 个数据集上平均性能提升了 22.76%，同时相比于 LTRWES 方法，本文的方法平均提升了 9.29%，这是因为相比于之前的噪音过滤方法，本文提出的基于生成模型的噪音过滤考虑了缺陷报告的语义信息，通过识别安全缺陷报告的语义区域实现非安全缺陷报告异常程度的准确判断，进而过滤掉可能是噪音的非安全缺陷报告. 另外，本文使用深度学习模型提取缺陷报告的深层语义信息，使得模型在未知数据上具有更强的泛化能力. 实验结果表明本文提出的基于生成模型的噪音过滤和深度学习相结合的安全缺陷报告识别方法优于传统的基于文本挖掘和机器学习的安全缺陷报告识别方法.

5.3　针对 RQ3 的结果分析

RQ3. 本文提出的噪音过滤方法相比于目前最先进的缺陷报告噪音过滤方法是否能更有效地提高基于深度学习的安全缺陷报告识别模型的性能？

目前在安全缺陷报告识别任务上对噪音过滤效果最好的方法是我们在前期工作中提出的基于内容相似度的噪音过滤方法 LTRWES，该方法首先使用 $BM25F_{ext}$ 计算每一组非安全缺陷报告和所有安全缺陷报告之间的内容相似度得分，然后使用不同的选择算法过滤可能是噪音的非安全缺陷报告. 该方法相比于基于安全关键词的过滤方法（即 FARSEC），在不同的数据集上的识别效果均得到了显著的提升. 但是 LTRWES 也存在较为明显的缺陷，即为了平衡安全缺陷报告和非安全缺陷报告在数量上的差距，该方法移除了大量的非安全缺陷报告使得训练集的数量大大减少，造成过滤之后的训练数据集虽然能够满足机器学习模型训练的需要，但是无法训练泛化能力很好的深度学习模型. 为了验证本文提出的基于生成模型的噪音过滤方法是否能更有效地生成适合深度学习模型训练的高质量数据集，本节将使用以往最先进的非安全缺陷报告噪音过滤方法

得到的在机器学习模型上取得最优效果的训练数据集传入基于深度学习的安全缺陷报告识别模型进行训练，并将模型在相同测试集上的实验结果作为本节基准方法的实验结果，与本文提出的基于生成模型的噪音过滤方法 FSDON 的效果进行比较，对比结果如表 11 所示.

表 11 不同噪音过滤方法结合基于深度学习的识别模型在不同项目上的最优性能(*g*-measure)对比

数据集	FSDON		FARSEC		LTRWES	
	深度学习模型	*g*-measure	深度学习模型	*g*-measure	深度学习模型	*g*-measure
Ambari	BiLSTM	97.61	BiLSTM	95.44	TextCNN	90.69
Camel	Multi-scale DCNN	73.10	BiGRU	61.85	Multi-scale DCNN	62.43
Chromium	GRU	89.69	BiGRU	87.74	GRU	81.47
Derby	Multi-scale DCNN	85.10	GRU	81.20	GRU	76.52
Wicket	Multi-scale DCNN	86.11	Multi-scale DCNN	76.40	Multi-scale DCNN	77.87

另外，为了更方便的比较不同的噪音过滤方法对基于深度学习的安全缺陷报告识别模型的训练过程产生的影响，本文以柱状图的形式描绘三种噪音过滤方法在所有数据集上的最优 *g*-measure 差异. 实验结果如图 6 所示. 从表 11 和图 6 中可知，本文提出的基于生成模型的噪音过滤方法(FSDON)在 5 个数据集上关于 *g*-measure 指标均达到了最优，并且相比于基于内容相似度的过滤方法(LTRWES)，本文的方法在 5 个数据集上平均性能提高了 8.52%，可能的原因有如下几点：

(1) 由于本文提出的噪音过滤方法使用分布式的词嵌入表示考虑了词的语义信息，并且利用生成模型从语料库中所有词的语义信息中识别可能的语义区域以及语义焦点，然后基于语义焦点准确地度量非安全缺陷报告包含安全缺陷的可能性. 因此相比于基于安全关键词(FARSEC)或者内容相似度(LTRWES)的过滤方法，本文方法可以充分挖掘缺陷的语义特征，从而提高了噪音过滤效果；

(2) 基于内容相似度的过滤方法(LTRWES)通过去除大量的非安全缺陷报告在传统基于机器学习的识别模型中取得了很好的效果，但是对于参数更多的深度学习模型而言，使用传统过滤方法得到的训练数据集的数量不足以训练泛化性能很好的深度学习模型，因此，基于内容相似度和基于安全关键词的过滤方法的性能要低于本文提出的基于生成模型(FSDON)的噪音过滤.

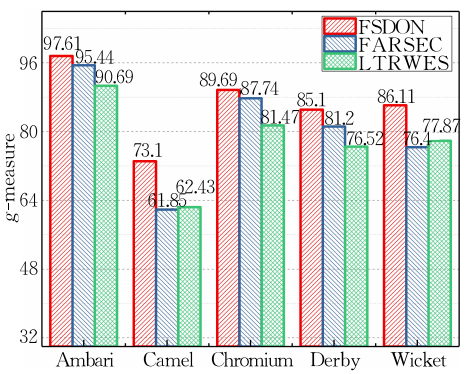

图 6 本文过滤方法与最先进的基于文本挖掘的过滤方法所生成的过滤数据集上训练的深度学习模型的最优性能(*g*-measure)对比

5.4 针对 RQ4 的结果分析

RQ4. 本文提出的方法是否能生成更有效的可疑安全缺陷报告列表？

对于安全缺陷报告识别任务，安全缺陷报告被正确识别的重要性要高于非安全缺陷报告被正确识别的重要性，这是因为如果安全缺陷报告一旦被错误的识别，将会给软件系统造成巨大的安全威胁. 因此，越来越多的研究更加关注预测模型在 *g*-measure 和 *recall* 指标上的表现. 例如 Peter 等人的 FARSEC 方法和 Jiang 等人的 LTRWES 方法都以 *g*-measure 指标作为机器学习模型优化的目标，即最大化预测模型在历史数据集上的 *g*-measure 值来调整机器学习模型的参数. 但是在最大化 *g*-measure 值的同时，也会使预测模型不可避免的产生大量的 *FP*(False Positives)实例(即非安全缺陷报告被预测为安全缺陷报告)，特别是对于非平衡数据集，这一问题尤其严重. 如果预测模型在正确识别出安全缺陷报告的同时，也产生了大量的误报实例，这无疑会增加开发人员或者维护人员的负担，因为大量预测为安全缺陷报告的实例仍然需要人工进行逐一确认. 为了解决上述问题，FARSEC 和 LTRWES 使用集成学习的思路，通过考虑多个预测模型的结果对识别为安全缺陷报告的实例进行排序，使得真正的安全缺陷报告应该更靠近列表的顶端.

为了检测本文提出的结合噪音过滤和深度学习的方法能否生成更有效的可疑安全缺陷报告列表，我们采用平均加权集成学习策略将不同的深度学习模型[28-30]组合成用于预测结果排名的元分类器. 具体步骤如下，首先选择具有最大 *g*-measure 和 *f*-measure 的深度学习模型作为用于集成学习的初级学习器. 其次，对于测试集中的每一个缺陷报告，本文使用所有初级学习器分别计算该缺陷报告

属于安全缺陷报告的概率，然后计算概率的平均值作为该缺陷报告被元分类器预测为安全缺陷报告的最终概率. 最后，根据预测的最终概率值按照从高到低对测试集中的所有缺陷报告进行排序.

考虑选择最大 g-measure 和 f-measure 的深度学习模型作为初级学习器的原因是为了平衡 TPR（真阳性率）和 FPR（假阳性率）的大小，缓解由于过渡追求 TPR 而造成预测模型产生大量的安全缺陷报告误报实例的问题，以确保大多数真实的安全缺陷报告更接近排序列表的顶端. 本文使用 MAP 指标评估我们提出的基于深度学习的集成模型的排序性能，实验结果如图 7 所示，其中 baseline 指测试集中

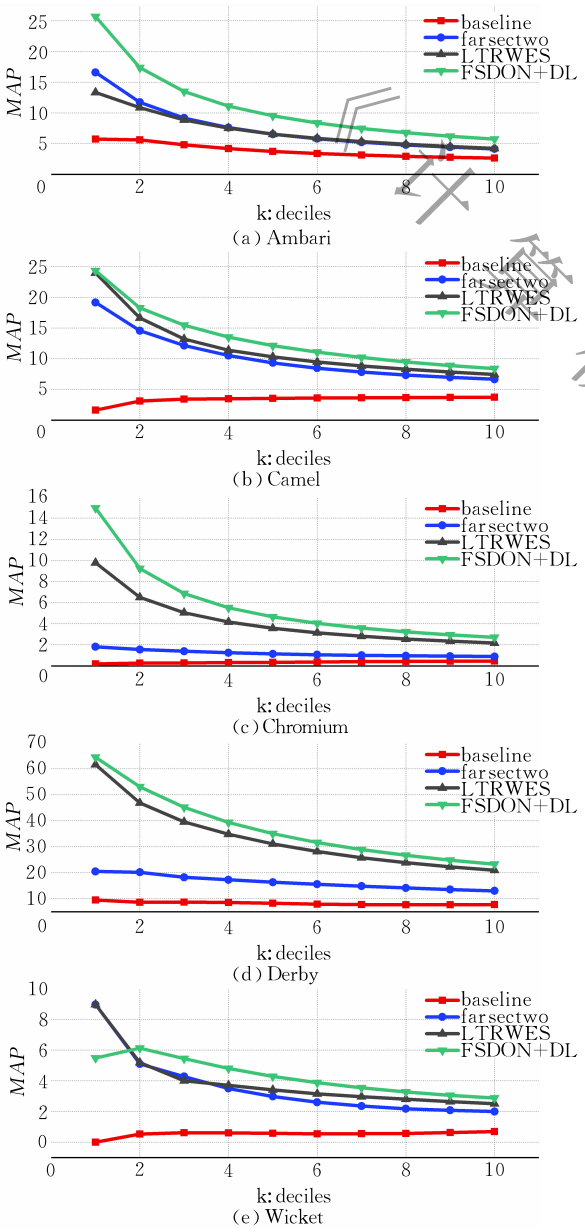

图 7　本文方法与基于文本挖掘和机器学习的方法在各个数据集上排序性能的对比

的缺陷报告按照时间提交的先后进行排序，并根据排序的结果计算的 MAP 值，其它线在图例中按先后顺序分别表示 FARSEC，LTRWES 和本文提出的方法在不同项目上的最优 MAP.

从实验结果可知，两种目前已知的最先进的方法（FARSEC 和 LTRWES）以及本文提出的方法（FSDON+DL）的排序性能都要明显优于 baseline，表明三种方法都能有效地预测测试集中的安全缺陷报告. 其中 LTRWES 由于使用基于内容相似度的过滤方法使得过滤之后的训练数据集的质量要高于基于安全关键词过滤方法（即 FARSEC）得到的数据集质量，因此 LTRWES 方法相比于 FARSEC 方法在大多数项目上都取得了更优的排序性能. 但是无论是 LTRWES 还是 FARSEC 在过滤阶段都没有考虑缺陷报告的语义信息，因此有可能因为一词多义或者多词同义的问题造成非安全缺陷报告的异常评估不准确，而本文通过使用基于 sHDP 的生成模型识别缺陷报告中可能的安全语义区域和安全语义焦点，可以更有效和更准确地评估非安全缺陷报告包含安全缺陷的可能性. 图 7 的实验结果表明本文提出的基于生成模型的过滤方法相对于其他方法更有效.

6　总　结

本文提出一种基于噪音过滤和深度学习相结合的安全缺陷报告识别方法，该方法首先利用词嵌入技术获取缺陷报告语料库中所有单词的分布式向量表示；其次使用 vMF 分布刻画词嵌入空间中向量的分布，并基于 sHDP 的生成模型划分所有可能的语义区域，在同一个语义区域中的词表达了相同的语义主题；然后利用和安全缺陷报告相关的语料库对语义区域进行词频统计，从而获得安全语义焦点；利用安全语义焦点计算非安全缺陷报告的异常分数，按照异常分数的高低对所有非安全缺陷报告进行排序并按一定比例过滤可能是噪音的非安全缺陷报告；将过滤之后的非安全缺陷报告和所有的安全缺陷报告组成新的训练数据集；最后利用不同的深度学习模型提取缺陷报告的深层语义特征，并完成安全缺陷报告的自动识别.

在 Ambari、Camel、Chromium、Derby 以及 Wicket 数据集上的实验结果表明，本文提出的方法明显优于目前最先进的基于文本挖掘和机器学习的方法以及优于以往单纯基于深度学习的安全缺陷报告识别方法. 实验不仅从分类和排序两个角度验证了本文提出的基于生成模型的缺陷报告噪音过滤方法在提

升缺陷报告识别模型性能上的有效性,同时也验证了基于深度学习的安全缺陷报告识别方法相比于传统的机器学习方法可以更有效地从复杂的文本内容中提取安全缺陷特征.在今后的工作中,我们会通过人工标注的方式在更多的实际项目和开源项目上生成安全缺陷报告数据集,并广泛验证本文提出的方法的有效性.另外,考虑部分项目可能含有较少的历史缺陷报告,在未来工作中我们也会对跨项目安全缺陷报告识别进行深入的研究.

参 考 文 献

[1] Wang Qing, Wu Shu-Jian, Li Ming-Shu. Software defect prediction. Journal of Software, 2008, 19(7): 1565-1580(in Chinese)
(王青,伍书剑,李明树. 软件缺陷预测技术. 软件学报, 2008, 19(7): 1565-1580)

[2] Williams L, McGraw G, Migues S. Engineering security vulnerability prevention, detection, and response. IEEE Software, 2018, 35(5):76-80

[3] Peters F, Tun T, Yu Y, et al. Text filtering and ranking for security bug report prediction. IEEE Transactions on Software Engineering, 2017, 45(6): 615-631

[4] Jiang Y, Lu P, Su X, et al. LTRWES: A new framework for security bug report detection. Information and Software Technology, 2020, 124: 106314

[5] Sun C, Lo D, Khoo S C, et al. Towards more accurate retrieval of duplicate bug reports//Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011). Lawrence, USA, 2011: 253-262

[6] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, 2013, 26: 3111-3119

[7] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781, 2013

[8] Kim Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014

[9] Wang S, Huang M, Deng Z. Densely connected CNN with multiscale feature attention for text classification//Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18. Stockholm, Sweden, 2018: 4468-4474

[10] Gegick M, Rotella P, Xie T. Identifying security bug reports via text mining: An industrial case study//Proceedings of the IEEE Working Conference on Mining Software Repositories. Cape Town, South Africa, 2010: 11-20

[11] Behl D, Handa S, Arora A. A bug mining tool to identify and analyze security bugs using naive Bayes and TF-IDF//Proceedings of the 2014 International Conference on Optimization, Reliability, and Information Technology (ICROIT). Faridabad, India, 2014: 294-299

[12] Goseva-Popstojanova K, Tyo J. Identification of security related bug reports via text mining using supervised and unsupervised classification//Proceedings of the 2018 18th IEEE International Conference on Software Quality, Reliability, and Security. Lisbon, Portugal, 2018: 344-355

[13] Pereira M, Kumar A, Cristiansen S. Identifying security bug reports based solely on report titles and noisy data//Proceedings of the 2019 IEEE International Conference on Smart Computing (SMARTCOMP). Washington, USA, 2019: 39-44

[14] Wu X, Zheng W, Chen X, et al. CVE-assisted large-scale security bug report dataset construction method. Journal of Systems and Software, 2019, 160: 110456

[15] Goldberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning. New York: Addison-Wesley, 1989

[16] Palacio D N, Mccrystal D, Moran K, et al. Learning to identify security-related issues using convolutional neural networks//Proceedings of the 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). Cleveland, USA, 2019: 140-144

[17] Zheng Wei, Chen Jun-Zheng, Wu Xiao-Xue, et al. Empirical studies on deep-learning-based security bug report prediction methods. Journal of Software, 2020, 31(5): 1294-1313(in Chinese)
(郑炜,陈军正,吴潇雪等. 基于深度学习的安全缺陷报告预测方法实证研究. 软件学报, 2020, 31(5): 1294-1313)

[18] Morrison P, Oyetoyan T D, Williams L. Identifying security issues in software development: Are keywords enough?//Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings. Gothenburg, Sweden, 2018: 426-427

[19] Joulin A, Grave E, Bojanowski P, et al. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759, 2016

[20] Pennington J, Socher R, Manning C. Glove: Global vectors for word representation//Proceedings of the Conference on Empirical Methods in Natural Language Processing. Doha, Qatar, 2014: 1532-1543

[21] Batmanghelich K, Saeedi A, Narasimhan K, et al. Nonparametric spherical topic modeling with word embeddings//Proceedings of the Meeting of the Association for Computational Linguistics. Berlin, Germany, 2016: 537-542

[22] Arfken G, Weber H, Harris F. Mathematical Methods for Physicists. Seventh Edition. San Diego: Academic Press, 2012

[23] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780

[24] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014

[25] Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278-2324

[26] Ohira M, Kashiwa Y, Yamatani Y, et al. A dataset of high impact bugs: Manually-classified issue reports//Proceedings of the 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories (MSR). Florence, Italy, 2015: 518-521

[27] Scandariato R, Walden J, Hovsepyan A, et al. Predicting vulnerable software components via text mining. IEEE Transactions on Software Engineering, 2014, 40(10): 993-1006

[28] Fumera G, Roli F. Performance analysis and comparison of linear combiners for classifier fusion//Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). Berlin, Germany, 2002: 424-432

[29] Fumera G, Roli F. A theoretical and experimental analysis of linear combiners for multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005, 27(6): 942-956

[30] Sewell M. Ensemble learning. RN, 2008, 11(2): 1-34

JIANG Yuan, Ph. D. candidate. His main research interests include mining software repository, software vulnerabilities detection, source code representation.

MU Chen-Guang, M. S. His current research interests include mining software repository and security bug report detection.

SU Xiao-Hong, Ph. D., professor. Her research interests include intelligent software engineering, software vulnerability identification, code representation learning, bug triaging and localization, clone detection and code search.

WANG Tian-Tian, Ph. D., associate professor. Her current research interests include software engineering, program analysis and computer aided education.

Background

During the whole-life cycle of software development and maintenance, it is inevitable that there will be various software bugs. The security-related software bugs are easy to be exploited by malicious hackers to attack the software-system if not fixed early, which may carry the risk of endangering human lives or financial damage. Therefore, security bug report (SBRs) detection belongs to the research area of software security. Although many methods have been proposed for solving this task, there are still remaining two challenges that need to be solved. The first challenge to overcome is how to design an optimal method to accurately reduce noise (i.e., mislabelled non-security bug reports) to improve the trained model performance. The second key challenge is how to capture deep semantic information from the text descriptions of bug reports. For example, the state-of-the-art methods (i.e., FARSEC and LTRWES) filter the noisy non-security bug reports (NSBRs), which have higher similarity with security bug reports, to generate high-quality training datasets. However, both of them do not take the semantic information of bug reports into account when calculating the outlier score for each NSBR, which may cause the filtered datasets to be inaccurate. Additionally, owing to the small sample size and complex characteristics of security-related bug reports, it is difficult for most previous work based on machine learning methods (i.e., FARSEC and LTRWES) to capture deep semantic information from textual fields of bug reports.

To address these challenges, we propose a novel framework to predict unknown security bug reports by combining semantic-based noise filtering with deep learning techniques. More concretely, it leverages the proposed FSDON method to filter the noisy NSBRs at a semantic level. Then, it builds automatic feature extractors and SBRs detection models to capture the semantics of bug reports via different deep learning networks. This method is evaluated on 5 different datasets, and the experimental results show that the g-measure performance of this method can be improved by 8.26% on average compared with the state-of-the-art methods (i.e., FARSEC and LTRWES). The research group has done much work in designing effective and efficient filtering and detecting algorithms, such as our previous model, LTRWES, presented in Ref. [4]. The work is supported by the National Natural Science Foundation of China (Grant No. 61672191), and "the 13th Five-Year" National Science and Technology Major Project of China (Grant No. 2017YFC0702204).