软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(3):1257−1279 [doi: 10.13328/j.cnki.jos.006812]

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

基于知识图谱的跨项目安全缺陷报告预测方法*

郑 炜[1,4,5], 刘程远[1], 吴潇雪[2], 陈 翔[3,6], 成婧源[1], 孙小兵[2], 孙瑞阳[1]

[1](西北工业大学 软件学院, 陕西 西安 710072)

[2](扬州大学 信息工程学院, 江苏 扬州 225127)

[3](南通大学 信息科学技术学院, 江苏 南通 226019)

[4](空天地海一体化大数据应用技术国家工程实验室 (西北工业大学), 陕西 西安 710072)

[5](大数据存储与管理工业和信息化部重点实验室 (西北工业大学), 陕西 西安 710072)

[6](信息安全国家重点实验室 (中国科学院 信息工程研究所), 北京 100093)

通信作者: 吴潇雪, E-mail: xiaoxuewu@yzu.edu.cn

摘 要: 安全缺陷报告可以描述软件产品中的安全关键漏洞. 为了消除软件产品的安全攻击风险, 安全缺陷报告(security bug report, SBR) 预测越来越受到研究人员的关注. 但在实际软件开发场景中, 需要进行软件安全漏洞预测的项目可能是来自新公司或属于新启动的项目, 没有足够的已标记安全缺陷报告供在实践中构建此软件安全漏洞预测模型. 一种简单的解决方案就是使用迁移模型, 即利用其他项目已经标记过的数据来构建预测模型. 受到该领域最近的两项研究工作的启发, 以安全关键字过滤为思路提出一种融合知识图谱的跨项目安全缺陷报告预测方法 KG-SBRP (knowledge graph of security bug report prediction). 使用安全缺陷报告中的文本信息域结合 CWE (common weakness enumeration) 与 CVE Details (common vulnerabilities and exposures) 共同构建三元组规则实体, 以三元组规则实体构建安全漏洞知识图谱, 在图谱中结合实体及其关系识别安全缺陷报告. 将数据分为训练集和测试集进行模型拟合和性能评估. 所构建的模型在 7 个不同规模的安全缺陷报告数据集上展开实证研究, 研究结果表明, 所提方法与当前主流方法 FARSEC 和 Keyword matrix 相比, 在跨项目安全缺陷报告预测场景下, 性能指标 *F*1-score 值可以平均提高 11%, 除此之外, 在项目内安全缺陷报告预测场景下, *F*1-score 值同样可以平均提高 30%.

关键词: 软件安全; 安全缺陷报告预测; 跨项目; 知识图谱; 领域知识

中图法分类号: TP311

中文引用格式: 郑炜, 刘程远, 吴潇雪, 陈翔, 成婧源, 孙小兵, 孙瑞阳. 基于知识图谱的跨项目安全缺陷报告预测方法. 软件学报, 2024, 35(3): 1257–1279. http://www.jos.org.cn/1000-9825/6812.htm

英文引用格式: Zheng W, Liu CY, Wu XX, Chen X, Cheng JY, Sun XB, Sun RY. Cross-project Prediction Method of Security Bug Reports Based on Knowledge Graph. Ruan Jian Xue Bao/Journal of Software, 2024, 35(3): 1257–1279 (in Chinese). http://www.jos.org.cn/1000-9825/6812.htm

Cross-project Prediction Method of Security Bug Reports Based on Knowledge Graph

ZHENG Wei[1,4,5], LIU Cheng-Yuan[1], WU Xiao-Xue[2], CHEN Xiang[3,6], CHENG Jing-Yuan[1], SUN Xiao-Bing[2], SUN Rui-Yang[1]

[1](School of Software, Northwestern Polytechnical University, Xi'an 710072, China)

[2](College of Information Engineering, Yangzhou University, Yangzhou 225127, China)

[3](School of Information Science and Technology, Nantong University, Nantong 226019, China)

[4](National Engineering Laboratory for Integrated Aero-space-ground-ocean Big Data Application Technology (Northwestern Polytechnical University), Xi'an 710072, China)

* 基金项目: 国家自然科学基金 (62202414, 62141208); 国家重点研发计划 (2020YFC0833105Z1)
收稿时间: 2022-01-06; 修改时间: 2022-06-26, 2022-08-19; 采用时间: 2022-09-29; jos 在线出版时间: 2023-07-05
CNKI 网络首发时间: 2023-07-06

[5](Key Laboratory of Big Data Storage and Management (Northwestern Polytechnical University), Ministry of Industry and Information Technology, Xi'an 710172, China)

[6](State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences), Beijing 100093, China)

Abstract: Security bug reports (SBRs) can describe critical security vulnerabilities in software products. SBR prediction has attracted the increasing attention of researchers to eliminate security attack risks of software products. However, in actual software development scenarios, a new company or new project may need software security bug prediction, without enough marked SBRs for building SBR prediction models in practice. A simple solution is employing the migration model, which means that marked data of other projects can be adopted to build the prediction model. Inspired by two recent studies in this field, this study puts forward a cross-project SBR prediction method integrating knowledge graphs, i.e., knowledge graph of security bug report prediction (KG-SBRP), based on the idea of security keyword filtering. The text information field in SBR is combined with common weakness enumeration (CWE) and common vulnerabilities and exposures (CVE) Details to build a triple rule entity. Then the entity is utilized to build a knowledge graph of security bugs and identify SBRs by combining the entity and relationship recognition. Finally, the data is divided into training sets and test sets for model fitting and performance evaluation. The built model conducts empirical research on seven SBR datasets with different scales. The results show that compared with the current main methods FARSEC and Keyword matrix, the proposed method can increase the performance index *F*1-score by an average of 11% under cross-project SBR prediction scenarios. In addition, the *F*1-score value can also grow by an average of 30% in SBR prediction scenarios within a project.

Key words: software security; prediction of security bug report; cross-project; knowledge graph; domain knowledge

随着信息技术的快速发展, 软件在人们日常生活中扮演着日益关键的角色, 它在促进了行业高速发展的同时, 也助长了不少重大事故的产生[1]. 根据全球权威公共漏洞披露机构 CVE [2,3]的统计表明, 近年来, 系统安全漏洞数量迅速增加, 2011 年发现的安全漏洞数目为 4 155 个, 但随着时间的增长, 到 2021 年上升到 16 601 个. 软件缺陷一般被称为 bug, 漏洞属于软件缺陷的一类, 它可以归类为软件安全方面的缺陷. 软件缺陷是指计算机软件或程序中存在某种破坏正常运行能力的问题、错误, 或者隐藏的功能缺陷. 漏洞是指软件在设计、实现、配置策略及使用过程中出现的缺陷, 它可能导致攻击者在未授权的情况下访问或破坏系统. 缺陷存在于软件架构和设计中, 漏洞存在于软件代码 (源代码或二进制) 中. 从效果看缺陷就是软件正常运行会出现故障的, 而漏洞是软件被攻击者攻击才会出故障的点. 二者被发现后会被记录在缺陷报告上传入缺陷报告管理系统等待维护人员解决. 缺陷报告作为描述软件运行过程中出现错误的重要载体, 能够为软件安全漏洞预测提供重要的信息来源, 是软件开发和维护过程中重要的组成部分. 然而某些缺陷报告描述了攻击者可能利用的安全漏洞, 如果这些漏洞在修复之前暴露在外, 就可能被恶意攻击者利用发起软件系统攻击, 造成系统运行崩溃、大量重要信息泄露以及企业巨额财产经济损失等严重后果, 对软件系统安全造成了极大的威胁[1]. 因此, 如何快速、准确、自动化地识别软件安全缺陷报告一直以来是信息安全和软件工程领域的研究重点和热点.

近年来, 在安全缺陷漏洞报告预测中很多工作都集中关注到项目内预测训练 (within-project prediction, WPP) 即选取一个项目的部分数据作为数据集进行预测模型构建和训练, 并用其他未使用的数据作为测试集利用训练好的模型对其进行检测和模型性能测试. 这些大多数是由监督学习来推动的, 在训练集不同的情况下, 相同模型的泛化能力较差. 而在现实的软件开发场景中, 要进行软件安全漏洞预测的很有可能是某个新启动的项目, 没有足够的本地数据存储库供在实践中构建此软件安全漏洞预测模型. 目前在跨项目软件安全缺陷报告预测 (transfer project prediction, TPP) 训练数据的收集过程中, 有两种方法: 第 1 种是借助一些软件系统衡量开发工具 (例如 Understand 工具), 能够比较方便地搜集到项目管理中程序模块的各类软件度量数据信息, 而当后续研究这个模型里面是否存在漏洞信息时, 则要求领域内的技术专家进行深入分析缺陷报告管理系统中的大量漏洞信息以及版本管理系统中的代码修改日志, 但是这种方式代价比较高昂而且容易将信息标记错误. 另一种有效的解决方案是利用迁移学习模型, 即使用其他项目已经训练过的数据来构建预测模型. 由于各个项目模块之间设计过程、主要目标领域、程序设计环境和编程语言以及工程项目的参与人员技术水平等并不相同, 所以检测的源项目和目标项目之间会存在巨大的数据分布差异. 目前因为传统深度学习模型的限制, 在跨项目预测上并未有效缓解源项目和目标项目之间

的数据信息分配差, 导致实际预测效果并不理想.

传统机器学习算法仅考虑单一属性对 SBR 进行预测, 同时滤掉了安全交叉词, 忽略 SBR 内部间的关系, 造成部分数据或关系遗漏, 从而导致模型预测效果不理想. 关系与属性是信息理解和认知的基石, 知识图谱作为一种语义网络, 可以将 SBR 可视化, 而构建知识图谱过程的本质就是将散乱的 SBR 数据整合起来, 形成 SBR 知识群. 帮助我们拥有更好的视觉体验, 更加深层次去理解各项数据之间的关系. 同时不同类型实体以不同颜色呈现, 还能帮助我们挖掘出错误造成原因以及出现源头等更多重要的信息. 相比传统方法知识图谱不仅可以通过关键词结合实体关系识别 SBR, 提升 SBR 识别的准确率, 同时存在重复实体时能压缩数据量存储数据. 为 SBR 领域的探索提供了新的思路.

故本文提出了一种通过构建了安全漏洞领域知识图谱来实现跨项目预测安全缺陷报告的 KG-SBRP 方法. 领域知识图谱已经成为人工智能时代的一个研究课题, 目前对知识图谱的研究主要集中在知识图谱的构造上. 文中我们利用 SBR 的相关数据构建软件安全漏洞知识图谱以提高 SBR 预测的准确性, 通过使用基于规则的命名实体识别来建立实体之间的关系, 以此构建知识图谱. 并且使用实体、实体间的关系及其安全短句对缺陷报告做出预测.

我们将构建好的软件安全漏洞知识图谱通过 KG-SBRP 方法应于 7 个数据集上进行跨项目 SBR 预测 (其中 5 个为图谱构建使用数据集[4−8], 另外新增了两个大型数据 Chromium_Large 和 Mozilla_Large 来验证本文方法在大型数据集上的有效性), 并且与 Peters 等人[9]提出的方法 FARSEC 进行比较. 结果表明, 本文提出的 KG-SBRP 方法对 SBR 预测能力, 要显著优于 FARSEC 方法, 在项目内预测上 $F1$-score 与 Peters 等人[9]和 Wu 等人[10]给出的方法相比平均提高 30%, 跨项目上 $F1$-score 平均提高 11%. G-measure 平均提高了 10%.

本文主要贡献可总结如下.

(1) 根据缺陷报告中对安全缺陷的文本描述的特征, 本文提出了一种融合知识图谱的跨项目 SBR 预测方法 KG-SBRP, 使用了实体+边的关系+短语词组相结合的跨项目 SBR 预测模型. 据我们所知, 我们是首次使用知识图谱结合软件安全漏洞领域知识来改进跨项目 SBR 预测性能的团队. 构建了 5 个数据集通用的软件安全漏洞知识图谱, 为该领域的研究提供了参考.

(2) 基于实际开源项目验证了本文所提出方法 KG-SBRP 的有效性, 实证研究共分析了来自 Ambari、Camel、Derby、Wicket、Chromium、Chromium_Large 和 Mozilla_Largez 这 7 个项目累积 234 601 个缺陷报告, 最终结果表明, KG-SBRP 方法性能要显著优于当前主流跨项目缺陷预测 FARSEC 方法. 本文最后深入分析了 KG-SBRP 方法是如何影响 SBR 预测的有效性、本文方法相对比传统方法的优势以及对 SBR 有效性影响因素分析.

本文第 1 节介绍跨项目安全缺陷报告的研究背景和相关工作. 第 2 节介绍软件安全漏洞知识图谱构建的具体过程和方法细节. 第 3 节介绍本文的实证研究, 包括研究问题、评测对象、评价指标、实验方法流程以及基准方法. 第 4 节对实证研究结果进行详细分析和总结. 最后总结全文, 指出本文工作不足和对未来的展望.

1 相关工作

近年来, 软件系统规模越来越大, 完全健壮的软件已经很难被设计实现, 伴随而来的软件系统安全问题日益突出, 为了保证软件的使用安全, SBR 预测也受到学术界和工业界越来越多的关注. 本文主要研究如何结合知识图谱进行 SBR 文本特征挖掘, 从而更加高效、准确地完成跨项目 SBR 预测. 本节将从项目内安全缺陷报告预测、跨项目缺陷预测和跨项目安全缺陷报告预测 3 个方面对现有工作进行说明.

1.1 安全缺陷报告预测

为了降低软件系统的安全风险, SBR 预测已成为当前软件工程领域的一个研究热点. 由于 SBR 检测被表述为一个二元分类问题, 大多数工作使用基于机器学习的文本挖掘方法来实现这一点, 因为缺陷报告的主要信息在字段描述中以文本格式描述. 目前对安全缺陷报告进行预测, 主要通过以下几个方面进行: 1) 首先提取缺陷报告中的重要信息 (目前文本信息提取较多). 2) 将 1) 中提取到的信息进行预处理清洗 (大小写转换, 词根还原等). 清洗过

后将文本通过词袋模型或词频-逆文本频率等文本表示模型将句子中的词转换成特征向量, 生成特征矩阵. 3) 再将这些词的特征向量同 SBR 进行余弦或者欧氏距离设置阈值进行判比, 根据结果修改训练模型. 4) 根据训练好的模型对测试集进行测试, 检测测试集的报告是否是安全缺陷报告.

近年来提出了许多基于机器学习的 SBR 预测方法[9−14]. 首次将文本分析技术应用于软件安全缺陷报告研究是的 Gegick 等人[15]. 他们试图基于关键词挖掘识别安全漏洞报告, 利用缺陷报告中的自然语言描述信息, 对 Cisco 真实项目的数据集进行实证研究. 随后 Wijayasekara 等人[16]对 Linux 内核和 MySQL 项目中暴露的漏洞进行了分析. 他们提出了一种通过从错误报告的摘要和描述中提取信息来识别漏洞的方法. 他们的结果表明经过分类器能够识别隐藏 bug. 但是在类不平衡方面处理不是很好, 且精度不够高.

Peters 等人[9]经研究发现一些非安全缺陷报告同样包含和安全相关的关键词, 我们称这样的词为“安全交叉词”. 这些带有安全交叉词的非安全缺陷报告对模型的训练来说, 相当于引入了噪音, 势必会影响我们模型的训练效果, 而这种不利的影响会在类别不平衡的情况下被扩大. 他们提出了一种用于 SBR 预测的噪声过滤的 FARSEC 方法. 在训练预测模型前将含有安全关键词的 NSBR 从训练集中移除, 从而提高模型对安全缺陷报告的识别效果. 然而, 该方法存在所提取的安全关键词不准确, 以及缺陷报告的向量表示稀疏的问题.

最近, Wu 等人[10]对 5 个公开的 SBR 预测数据集的标签正确性进行了全面审查, 发现这些数据集中存在大量错误标签, 这可能会误导 SBR 预测的研究方向. 他们通过手动分析每个错误报告并且通过比较分类模型来纠正数据. 结果表明, 经过清理的数据集提高了分类模型的性能, 比 Peters 等人[9]和 Shu 等人[12]提出的方法在干净数据集上的性能要比在噪声数据集上的性能好得多. 此外, 对于干净的数据集, 简单的文本分类模型可以显著优于 Peters 等人[9]和 Shu 等人[12]采用的基于安全关键字矩阵的方法.

1.2 跨项目缺陷预测

跨项目缺陷预测是这些年的热门话题研究之一[17−19]. 与安全缺陷报告预测不同, 研究人员在实际开发场景中发现, 需要进行软件安全漏洞预测的有时候可能是一个新公司或新项目, 其历史缺陷数据是非常稀缺的, 并且软件开发方法和语言的更新迭代十分迅速, 如果只使用同一个项目的历史缺陷数据用于训练, 往往很难构建有效且实用的缺陷预测模型以用于软件质量保障过程. 因此研究人员提出了跨项目软件缺陷报告预测, 大多数跨项目缺陷预测方法有一个重要的前提, 即它们需要源项目和目标项目具有相同的软件度量, 而摆脱了该限制的异构缺陷预测技术近年来引起了大量的研究兴趣.

根据预测的场景不同, 目前将已有的跨项目迁移学习方法分为 3 类, 分别为有监督学习、无监督学习和半监督学习方法. 有监督学习方法是目前研究人员最热衷的方法, 它主要基于训练集来构建模型进行预测. Nam 等人[20]利用迁移成分分析, 保留了源数据和目标数据部分重要的数据属性, 使得二者有部分相同的数据分布. Turhan 等人[21]提出 Burak 过滤法, 他们计算每个目标项目与训练集的欧氏距离, 选取距离最近的 K (K=10) 加入最终训练集, 缓解二者的数据分布差异, 但效果仍不理想. 无监督学习使用测试集中的程序模块进行预测, Wu 等人[22]提出了一种半监督字典学习方法, 他们利用这种技术对源数据进行打标签处理, 然后将处理过后的源数据和目标项目一起进行训练, 以此缓解分布差异. 而半监督学习会综合使用上述两种方法来构建模型预测. Zhong 等人[23]通过聚类的方法, 选择特定模块, 再邀请专业人员利用辅助信息进行标注. 本文拟采用有监督学习的跨项目缺陷预测方法, 以期能够客观的评价 KG-SBRP 方法的性能.

1.3 跨项目安全缺陷报告预测

目前, 随着科学技术的发展针对跨项目安全缺陷报告预测问题的研究有了越来越丰富的理论和实证研究, 同时也广泛地应用在各个领域. 软件规模的逐渐扩大, 软件安全漏洞数量也急剧上升, 如何准确高效的预测出 SBR 逐渐成为信息安全和软件工程领域的研究重点和热点, 而现有的 SBR 预测模型大多数都基于机器学习方法且集中于项目内 SBR 预测, 因此跨项目缺陷预测在软件信息安全领域的应用, 引起了研究人员的关注.

Peters 等人[9]在跨项目安全缺陷报告预测研究上, 他们认为候选训练集中也包含了许多与安全缺陷报告信息相关的实例, 筛选出候选训练集中特定的报告, 对预测有一定帮助. 因此他们提出了 Peters 过滤法, 具体来说就是

首先对候选训练集中的每个实例, 从目标项目中识别出与之最相似的实例并进行标记, 随后对目标项目中已标记的实例, 从候选训练集中选出与之最相似的实例作为特征, 添加到训练集中一起训练. 结果显示, 他们的方法在一定程度上可以提升 TPP 的预测性能.

本文将领域知识图谱同迁移学习模型相结合, 引入到 SBR 预测领域, 进行跨项目安全缺陷报告预测. 由于不同缺陷报告项目之间数据分布的差异性大、安全特征稀缺、SBR 预测数据中正样本 (SBR) 较少导致类别不平衡等问题 (例如在 Peters 等人[9]搜集的 5 个数据集中, 其正样本数目所占比例为 0.5%–9.0%), 本文采用了核心实体注入, 减少数据之间分布的差异性, 并使用实体联合关系检测与词组检测等方法相结合来预测跨项目安全缺陷报告, 更加严格的条件检测来提高识别的准确率.

2 本文提出的 KG-SBRP 方法

2.1 方法整体框架

本文提出 KG-SBRP 方法进行 SBR 预测的框架如图 1 所示, 其中包含 4 个阶段.

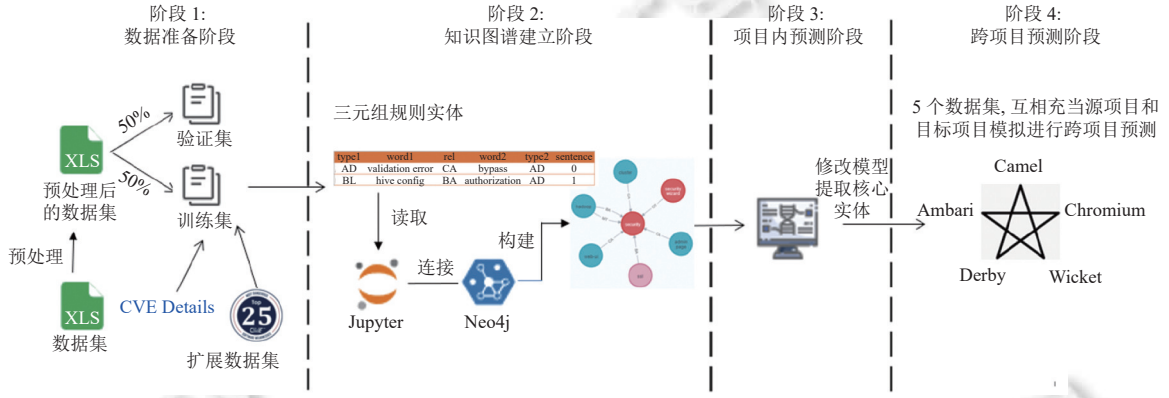

图 1 KG-SBRP 方法框架图

(1) 阶段 1: 数据准备阶段, 将本文所用的所有数据集进行基本自然语言处理, 然后统一分为两个部分, 标号前 50% 作为训练集进行模型的训练, 后 50% 作为测试集对训练的模型性能进行评估, 并使用安全漏洞领域知识 CWE Top25 最危险漏洞类型和 CVE Details 分别对训练集进行扩充.

(2) 阶段 2: 安全漏洞知识图谱构建阶段: 我们通过阶段 1 形成的结合领域知识的训练集, 对其进行基于三元组的规则实体建立. 对建立好的所有实体通过 Jupyter 工具读取, 连接 Neo4j 图数据库, 使用 Python 模块在图数据构建成安全漏洞知识图谱.

(3) 阶段 3: 项目内预测节点, 对构建好的知识图谱, 对阶段 1 中分分配的测试集进行项目内预测, 获取不同数据集的特征等, 根据实验结果调整实体、实体关系和检测阈值等, 为跨项目预测做准备.

(4) 阶段 4: 对阶段 3 中调整好的最优模型, 将本文所使用的 5 个数据集, 互相充当源项目和目标项目进行跨项目 SBR 预测.

2.2 数据集预处理

我们提取出缺陷报告数据集中的文本信息域: Summary 和 Description 作为语料预选库. 为了有效提高本文方法的准确率, 我们需要对语料库做一个预处理. 主要包含以下几个方面.

(1) 脏数据处理. 处理语料库的脏数据, 例如 Summary 和 Description 为空的缺陷报告、不是以英文作为描述文本的报告或乱码描述等情况. 以免对我们的测试结果造成影响.

(2) 单词大小写处理. 在不同的位置英文单词大小写不同好, 如苹果一词放在句子头为 Apple, 句子中则为 apple, 但二者表达意思一样. 为了不影响 KG-SBRP 方法的计算结果, 我们将英文单词大小写统一转成小写形式.

(3) 符号处理. 符号不携带任何语义上的信息, 故 KG-SBRP 方法在检测安全缺陷报告时只使用了文字信息而没有考虑文中符号. 故在预处理步骤我们将感叹后、句号、分号等符号去除, 以免对安全缺陷报告的检测产生影响.

(4) 词根还原. 词语的形态不同, 表达的语义却是一致的. 例如 walked 和 walking, 虽然是 walk 的不同形式, 但是表达的意思大致相同. 其次对我们在知识图谱中检测影响不大, 我们只考虑单词形式上相同, 而不去考虑语义的相似性. 故将两者还原为 walk. 本文采用 Python 的 NLTK 库进行词根还原.

(5) 停词处理. 由于 KG-SBRP 方法主要检测缺陷报告中的漏洞类型和安全关键词, 其他对于 KG-SBRP 方法检测的贡献很小, 并且会增加检测时间压力, 带来噪音. 故在预处理操作步骤进行停词处理.

经过数据预处理步骤后, 缺陷报告对比呈现表 1 所示.

表 1 预处理前后的文本对比

文本编号	文本信息	预处理后的文本
A	When I opened the remote revision files it default text edit	open remote revision files default text edit
B	Users can not find the reader’s files correctly	use find read file correct

2.3 数据源准备

构建我们的 SBR 知识图谱第一步就是要为实体的提取提供相关领域知识库. 在本研究中, 我们有 3 个来源, 即我们的目标项目、CWE 前 25 最危险漏洞类别和 CVE Details.

目标项目: 本研究的目标项目就是上文我们提到的 5 个基本数据集, 实践证明这 5 个开源项目均适用于 SBR 预测[9,12,14,24]. Wu 等人[10]纠正了这些数据集的错误标签, 并将它们视为基本事实.

本文遵循 Peters 等人[9]的工作, 将每个数据集分成两个相等的部分 (即 50% 和 50%). 我们将使用第 1 部分生成语料库, 另一部分作为测试集, 用于评估基于知识图谱的 SBR 预处理方法的有效性. 表 2 显示了 Ambari 项目中标为 26 的缺陷报告示例, 与以前的研究类似, 我们仅使用 Summary 和 Description 字段, 这是缺陷报告中信息量最大最多的信息字段, 并用文本进行了描述.

表 2 缺陷报告实例

Case	Details
Issue ID	26
Status	Verified (Closed)
Summary	Init Wizard: advanced Config validation errors can be bypassed
Description	Make the Naigos password (and re-type password) different so as you cause a validation error This will let the user move on to the next screen by ignoring all other validation errors on this page
Component	site
Priority	Major
Security	1
Reporter	Arpit Gupta
Created	2012/05/15 23:13:18 +0100
Assigned	2012/05/17 22:20:56 +0100

CWE 前 25 名最危险漏洞类别: CWE 是社区开发常见的软件缺陷列表, 如果 CWE 中的缺陷漏洞没有得到及时解决, 可能会导致我们多么系统非常容易受到攻击. 我们使用 CWE[25]在 2021 年报告中按普遍率和严重程度排名的 25 个最危险 CWE 类别作为我们领域知识库的另一个来源, 因为这些漏洞类别显示了可能出现严重软件漏洞的最广泛和最关键的编程错误[26]. 表 3 为 2021 年 CWE Top25 漏洞类型列表展示.

表 4 列出了 CWE 具体类别实例. 为了对安全相关领域知识进行更深入的探索, 我们不仅使用名称, 还使用每个 CWE 的描述和扩展描述作为数据源, 因为这些字段简洁地描述了每个 CWE, 并为我们提供了高质量的领域知识提取源. 表 4 提供了 CWE 类别的主要文本信息示例. 如表 4 所示, “Description”简要描述了 CWE 的特征, 而扩展描述提供了更多详细信息, 如 CWE 漏洞产生的根本原因和后果.

表 3 2021 年 CWE 前 25 名最危险类别

排名	ID	名称
1	CWE-787	Out of Bounds Write
2	CWE-79	Improper Neutralization of Input During Web Page Generation (“Cross-site Scripting”)
3	CWE-125	Out of Bounds Read
4	CWE-20	Improper Input Validation
5	CWE-78	Improper Neutralization of Special Elements Used in an OS Command (“OS Command Injection”)
6	CWE-89	Improper Neutralization of Special Elements Used in an SQL Command (“SQL Injection”)
7	CWE-416	Use After Free
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory (“Path Traversal”)
9	CWE-352	Cross-site Request Forgery (CSRF)
10	CWE-434	Unrestricted Upload of File with Dangerous Type
11	CWE-306	Missing Authentication for Critical Function
12	CWE-190	Integer Overflow or Wraparound
13	CWE-502	Deserialization of Untrusted Data
14	CWE-287	Improper Authentication
15	CWE-476	NULL Pointer Dereference
16	CWE-798	Use of Hard Coded Credentials
17	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
18	CWE-862	Missing Authorization
19	CWE-276	Incorrect Default Permissions
20	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor
21	CWE-522	Insufficiently Protected Credentials
22	CWE-732	Incorrect Permission Assignment for Critical Resource
23	CWE-611	Improper Restriction of XML External Entity Reference
24	CWE-918	Server-side Request Forgery (SSRF)
25	CWE-77	Improper Neutralization of Special Elements Used in a Command (“Command Injection”)

表 4 CWE 类别实例

Case	Details
Weakness ID	787
Abstraction	Base
Structure	Simple
Description	The software writes data past the end, or before the beginning, of the intended buffer.
Extend Description	Typically, this can result in corruption of data, a crash, or code execution. The software may modify an index or perform pointer arithmetic that references a memory location that is outside of the boundaries of the buffer. A subsequent write operation then produces undefined or unexpected results
ChildOf	119
ParentOf	121, 122, 123, 124

CVE Details 即公共漏洞和暴露细节, 是公开披露的网络安全漏洞列表. IT 人员和安全研究人员查阅 CVE 获取漏洞的详细信息, 进而确定漏洞的评分和优先级. 本文选取了安全漏洞领域 CVE Details 近 20 年高频漏洞 (即 20 年内该漏洞发生概率占比超过 10%) 作为数据集的扩展, 总共有 4 种漏洞类型, 分别为 Denial of Service, XSS, Execute Code, Overflow.

2.4 知识图谱的构建

2.4.1 实体语料库的生成

构建软件安全漏洞知识图谱是为软件安全缺陷报告自动化检测奠定基础, 安全缺陷报告的判断需要一定准确性, 如果出现误差就会导致人力物力的浪费. 且在软件安全领域知识图谱方面, 这快的研究资料有所欠缺, 通过自动或者半自动的方法构建知识图谱的方法在软件安全漏洞知识图谱上无法确实有效的施展, 所以我们通过人工标

注的方式构建实体和实体之间的关系.

软件工程中现有的大多数标签的生成都是基于文本的. 为了构建 SBR 知识图谱, 我们需要用我们的源数据来生成一个词汇语料库. 首先, 在对文本语料库进行预处理后, 对两个不同来源 (即目标项目和 CWE) 的文本进行分割, 得到部分单词. 我们使用 spaCy 工具进行文本标记化和单词词性还原[11,15,16,27]. 然后我们手动标记抽取的实体. 本文主要使用 SBR 中的文本信息域, 由于目前知识图谱在软件安全领域类的应用较少, 同时缺少实体标签数据. 故本文首先考虑了安全缺陷报告的特点, 仔细阅读每个安全缺陷报告的文本信息、理解其含义, 并结合 CWE 软件安全漏洞分类特性和查询相关资料后[10,14], 通过基于本文软件安全漏洞知识图谱的需要我们将实体大致分为以下几种不同的类别, 如软件名称、安全相关词语、缺陷位置、缺陷类型和其他词语或短语等几类. 例如 Ambari 数据集中编号为 3119 的句子: NullPointerException thrown while retrieving Ganglia properties, 结合上文提出的方法并对句子进行分析, NullPointerException 意为空指针异常是一种常见的安全漏洞类型, 将其标记为 VT. 而 Ganglia 是一个开源集群项目, 该句说明了在数据检索时在 Ganglia 出现了空指针异常, 故将 Ganglia 标记为 LOV. 表 5 列出了实体类别的类别类型、缩写和示例.

表 5 实体类型

实体类型	实例	解释
VT (vulnerability type)	SQL-injection, XSS	漏洞类型
SRW (safety-related words)	npe, memory	安全相关词
LOV (location of vulnerability)	HDP-Nagios, server-agent	漏洞出现位置
EA (execute action)	requese HTTP, create interface	执行某种操作
CP (component)	PHP, Ganglia, Chromium	编程语言, 类名等
OC (other category)	user, show	其他类型

由于本文的目的是进行 SBR 预测, 因此实体 VT (错误类型) 和 SRW (安全相关词), 作为我们着重要关注的点. 单词标记的准确性也将直接影响本文方法的有效性和所构建知识图谱的质量. 例如, 在某些句子中, 空指针解引用可能被写成 NPD (null pointer dereference), 如果没有足够安全领域知识储备的缺陷报告报告者可能会将这个词作为非安全词去报告. 然而, 空指针解引用是 CWE 前 25 名之一. 此外, 短语识别也是一个大问题. 例如, 在短语验证错误中, 单词 validation 与安全性无关, 但 validation error 就被视为错误类型需要标记. 还有例如 SQL 和 injection 分开来看毫无关系, 但是合在一起 SQL-injection 就是一种极为关键的错误类型. 本文创建了一种实体和模式列表将其命名为实体规则, 将有关系的单词联系在一起进行检测和抽取, 确保我们工作的有效性. 部分实体规则如表 6 所示.

表 6 部分实体规则

实体	实例
Validation error	{“label”: “VT”, “pattern”: [{“LOWER”: “validation”}, {“LOWER”: “error”}]}
SQL injection	{“label”: “VT”, “pattern”: [{“LOWER”: “SQL”}, {“LOWER”: “injection”}]}
Memory leak	{“label”: “VT”, “pattern”: [{“LOWER”: “Memory”}, {“LOWER”:“leak”}]}

为了确保分类结果的正确性, 本文采用“卡片分类法”进行样本标记, 即选取了 3 个彼此都有实体标记方面的经验, 并具有足够的软件安全漏洞知识的人员来做这项工作. 对于所有样本, 由 3 名标记人员分别完成标记, 最后比较标记结果. 我们通过计算标记结果的 Fleiss Kappa 值来度量标记结果的一致性, 所有数据标记结果的 Fleiss Kappa 平均值为 0.732, 表明标记结果具有高度一致性. 对于标记结果完全相同的记录, 其标记结果将直接作为最终标记结果; 对于标记结果存在不一致的情况, 由 3 名标记人员共同分析讨论, 决定最终标记结果用于构建知识图谱.

我们手动注释并创建一组在 spaCy 管道组件中定义的名为 EntityRuler 的规则, 以识别测试集缺陷报告中的

实体, 然后使用该规则的 to_disk 函数将定义的规则保存到以换行符分隔的 JSON 文件中, 方便我们检测使用. 表 6 pattern 中两个单词分开来写代表检测时二者是一个整体出现, SQL-injection 是 CWE Top 25 中非常重要的安全词, 在检测中是以 SQL-injection 形式出现检测, 这样基于安全词检测的效率就会大大提升, 而是不是割裂开看的 SQL 和 injection 进行检测, SQL 是用于访问数据库的标准计算机语言, 而 injection 代表注射, 跟安全词有一定差距.

2.4.2 实体关系建立

在第 2.4.1 节中, 我们进行文本标记化、单词词性还原和实体规则的建立. 在本节中, 我们添加实体之间的关系. 我们结合安全缺陷报告的特征, 在句子中寻找主语和修饰主语的形容词, 并根据主语和形容词之间的介词确定关系. 通过查询相关资料[28], 我们进行了手动检查并定义实体之间的关系. 表 7 展示了我们定义实体间的主要关系. 例如 Ambari 数据集中编号为 3 的句子: sentence custom config page: don't allow form submission if there are client side validation errors. 根据第 2.4.1 节中的标记过程, 我们将 form submission and validation errors 标记为 LOV. 通过阅读句子, 我们找到了介词 if, 因此, 我们判断这两个实体之间的关系是因果关系, 因此将这种关系标记为 BO (因果关系). 最后, 我们构建了每一对实体之间{entity, relationship, entity}的完整性, 比如{form submission, LOV, validation errors}.

表 7 实体关系

实体关系	解释
PGS (progressive)	递进关系: 先解决A才能解决B
BO (because of)	因果, 由于关系: 由于A所以导致B
SL (same level)	同类同级关系: A和B同为类名
LT (lead to)	导致, 造成关系: 由于A导致B (A为某种操作或者缺陷等)
CT (contain)	包含关系: A中存在B (B为缺陷类型)
SP (specify)	行为指明关系: 某些操作可能导致出现安全问题
OR (other relationship)	其他关系: 不属于上述关系即为其他

三元组用于标记每个与安全相关的句子、特征词类型及其关系. 然后, 我们将具有安全特征的词汇写入一个安全词汇表. 安全相关词汇表被构造到 JSON 文件中 (Neo4j 数据以 JSON 文件作为识别输入格式). 表 8 显示了最终生成的部分实体语料库. 将单词和相应的标签放在一起, 然后识别 SBR. 当存在相关的单词和短语时, 它可以自动识别关系短语.

表 8 基于规则的实体识别语料库

编号	语料库
1	{"label": "VT", "pattern": [{"LOWER": "encryption"}]}
2	{"label": "VT", "pattern": [{"LOWER": "virtual"}, {"LOWER": "memory"}]}
3	{"label": "BL", "pattern": ["LOWER": "node", "LOWER": "assignment"]}
4	{"label": "BL", "pattern": ["LOWER": "no", "LOWER": "server-side", "LOWER": "validation"]}
5	{"label": "SRW", "pattern": [{"LOWER": "insecurely"}, {"LOWER": "load"}]}
6	{"label": "VT", "pattern": [{"LOWER": "nullpointerexceptions"}]}
7	{"label": "SRW", "pattern": [{"LOWER": "SQL"}, {"LOWER": "injection"}]}

通过对复杂文档数据的有效处理和集成, 本文将安全缺陷报告转化为清晰明了的三元组数据. 在知识图谱中, 如果两个节点之间存在关系, 则它们将由无向边或有向边连接. 然后这个节点称为实体, 它们之间的边称为关系. 软件安全漏洞知识图谱[29−31]是指从软件安全漏洞领域获取的领域术语构建而成的知识图谱. 所构造的过程类似于一般知识图谱. 在软件安全知识图谱中, 我们只关注软件安全相关术语之间的关系. 在 SBR 预测领域, 我们可以使用知识图谱通过实体之间的联系建立 SBR 之间的关系. 建立好的部分三元组规则实体如表 9 所示.

表 9 建立好的部分三元组规则实体

type1	word1	rel	word2	type2
CP	ssl	PGS	security	SRW
SRW	security wizard	OR	fail	SRW
LOV	https	LT	fail	SRW
VT	nullpointerexception	PGS	deserialize	SRW
SRW	memory	BO	leak	VT
VT	stack	BO	overflow	VT
EA	https	LT	npe	SRW
LOV	cxf	CT	nullpointerexception	VT
SRW	exception	OR	error	VT
CP	trigger	CT	npe	SRW

2.4.3 知识存储

本文选择了 Neo4j 图数据库来储存我们的三元组规则实体, 将上文的定义的实体表示转化为 Neo4j 的节点, 实体之间的关系定义为边. 通过一个 Python 模块完成知识图谱的转换, 通过这个模块将所有的实体类和关系类转换为知识图谱的节点和边保存在 Neo4j 中.

图谱实例如图 2 所示, 不同类型的实体在知识图谱中有不同的颜色进行区分, 绿色部分是漏洞类型, 粉色部分是漏洞出现的位置, 黄色部分是造成出现漏洞的操作, 以此获得安全词汇表相关知识图谱. 从知识图谱中, 我们可以得到每个安全词汇之间的关系, 因此我们可以通过词汇与词汇本身的关系来预测 SBR. 在本文的实验中, 主要使用两种类型的安全缺陷词汇, SRW 和 VT. 例如 Ambari 数据集中编号为 1 的缺陷报告描述“Security wizard: phpIPAM 1.4 allows SQL injection via the app/admin/custom-fields/edit-result.php table parameter when action=add is used.”, 我们发现当单词 security、wizard、SQL 和 injection 同时出现在缺陷报告的描述文本中时, 通过与知识图谱比较, 可以一定程度上确定该缺陷报告是安全相关的, 即是一个 SBR.

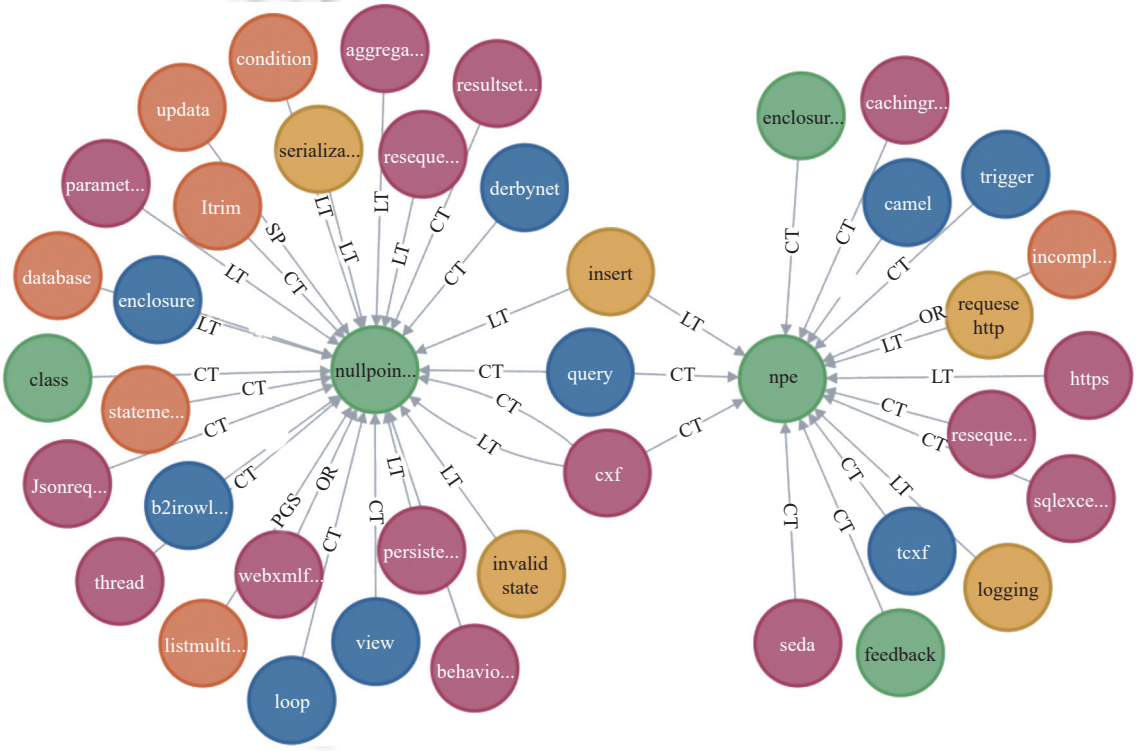

图 2 安全漏洞知识图谱实例

在知识图谱中, 当某一部分聚集实体越多, 连接边越多, 就说明该部分的安全领域知识越充足, 对某一个或者一类安全缺陷的特征信息描述越准确. 那么当有新的检测实体映射到时, 它是 SBR 的概率就会增加. 另外, 我们再根据节点之间的关系来判断, 这样我们就可以确定它是否为 SBR.

通过 Neo4j 数据库 Python 驱动模块的 Py2neo 操作, 形成了如图 2 所示的软件安全漏洞知识图谱实例. 如图 2 所示可知, 存在一种安全漏洞类型的两种不同表示 npe 和 nullpointexception, 它们存在于 Apache 的框架项目 cxf 中, 因此我们为 cxf 与 npe 和 nullpointexception 之间建立包含关系, 用 CT 表示. 在进行 request http 操作时也会导致 npe 的出现.

3 实验设计

为了支持本文的结论, 我们设计了如下两个研究问题 (research question, RQ) 来指导我们的实验设计.

RQ1: KG-SBRP 方法与已有基准方法相比是否可以提升跨项目安全缺陷报告预测性能?

RQ2: KG-SBRP 方法在跨项目安全缺陷报告预测性能上与传统深度学习方法相比有何优势?

3.1 评测对象

我们使用与 Peters 等人[9]和 Wu 等人[10]相同的 5 个数据集, 即 Ambari、Camel、Derby、Wicket、Chromium[4−8], 前 4 个项目来自 Web 服务器软件 Apache, 后者来源于 Google 的 Chrom 浏览器, 这 5 个数据集由 Wu 等人清理和共享[10], 以验证 KG-SBRP 方法的有效性和效率, 这次数据集纠正了错误标签, 可被认为是基本事实. 此 5 个数据集也是 SBR 预测中广泛应用的 5 个开源项目. 除此之外, 由于实际的软件项目中通常包含大量的缺陷报告, 为了验证本方法在实际的大数据集的安全缺陷报告预测效果, 又在原来的 5 个数据集的基础上, 增加了两个规模更大的数据集, 即 Chromium_Large 和 Mozilla_Large 数据集. Chromium_Large 同样来自于 Google 浏览器, 而 Mozilla_Large 数据集来源于 Mozilla 基金安全咨询会, 他们通过 Bugzilla 跟踪错误. 需要注意的是, 本实验中有两个数据集来自 Chromium 项目, 分别称作 Chromium_Large 和 Chromium, 其中 Chromium 是和文献 [14] 相同的数据集, 而 Chromium_Large 是本实验增加的规模更大的数据集. 数据集信息如表 10 所示.

表 10 安全缺陷报告数据集信息

数据集	时间	缺陷报告数量	安全缺陷报告数量	安全缺陷报告所占比例 (%)
Ambari	2004-09-26–2014-09-17	1 000	56	5.60
Camel	2007-07-08–2013-09-18	1 000	74	7.40
Derby	2004-09-28–2014-09-17	1 000	179	17.90
Wicket	2006-10-20–2014-11-09	1 000	47	4.70
Chromium	2008-08-30–2010-06-11	41 940	808	1.93
Chromium_Large	2008-09-02–2019-01-17	129 636	3 857	2.4
Mozilla_Large	2002-03-03–2019-01-17	59 025	1 445	2.9

3.2 评价指标

为了客观起见, 我们使用与 Peters 等人[9]和 Wu 等人[10]相同的 5 个指标对本文方法进行评估, 即召回率 (即他们工作中的 *pd*)、误报率 (*pf*)、精确率 (precision)、*F*1-score 和 *G*-measure. 在 SBR 预测的背景下, 召回率衡量从目标项目的总 SBR 中确定的真实 SBR 的百分比, 精确率衡量真实 SBR 在总确定 SBR 中的百分比. 误报率衡量 NSBR 误报为 SBR 的概率, 因此三者都是 SBR 预测的重要指标. *F*1-score 是召回率和准确度的调和平均值, 兼顾考虑了召回率和精确率, 可以用来评估准确度 (召回率) 的增加是否超过召回率 (准确度) 的减少[32]. *G*-measure 兼顾考虑了召回率与误报率. 由表 10 可知, 本文所用数据集存在严重的类别不平衡问题 (正样本数量所占比例仅为 0.5%–9.0%), 仅用单一指标作为度量标准存在很大问题, 会使预测结果浮动较大同时无法客观地评估预测模型的性能, 为了进行客观比较, 本文结合安全缺陷报告预测方面相关研究[3,11,18], 同时使用 *F*1-score 与 *G*-measure 作为实验结果主要关注点和评估指标[10,33−35].

在 SBR 预测的情况下, 每个错误报告的可能预测结果可以是 1 (SBR) 或 0 (NSBR). 预测结果的混淆矩阵如表 11 所示.

表 11 混淆矩阵

实际结果	预测结果	
	SBR	NSBR
SBR	*TP*	*FN*
NSBR	*FP*	*TN*

根据 KG-SBRP 模型对缺陷报告进行安全性预测, 确定为安全报告, 标记为 1, 非安全报告, 标记为 0. 给定一个 BR, 其预测可能结果有 4 种, *TP* (true positive) 真阳性, 表示预测为 SBR, 实际为 SBR, 预测正确; *FN* (false negative) 假阴性, 表示预测为 SBR, 实际为 NSBR, 预测错误; *FP* (false positive) 假阳性, 表示预测为 NSBR, 实际为 SBR, 预测错误; *TN* (true negative) 真阴性, 预测为 NSBR, 实际为 NSBR, 预测正确.

基于上述介绍, 我们依次介绍本文所使用的评价指标.

召回率 (*Recall*), 又叫查全率. 即检测到的 SBR 占实际总 SBR 数量的比例. 其计算公式如下:

$$Recall = \frac{TP}{TP+FN} \tag{1}$$

误报率 (probability of false alarm, *pf*): 在所有的 NSBR 预测中检测出 SBR 的概率, 即将 NSBR 误报为 SBR 的概率. 计算公式如下:

$$pf = \frac{FP}{FP+TN} \tag{2}$$

精确率 (*Precision*), 又叫查准率: 即准确检测到的 SBR 数量占所有 SBR 数量的比例, 其计算公式如下:

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

*F*1-score: 精准率和召回率都可以反映出模型检测的性能, 但是二者实际来看却是一队相反的指标, 精准率高说明模型在预测 SBR 时比较谨慎, 而召回率高, 说明模型倾向于将一个样本认为是正样本. 综合考虑精准率和召回率两个指标, 我们使用 *F*1-score 来评价模型实验性能, 计算公式如下:

$$F1\text{-score} = \frac{2\times Recall\times Precision}{Recall+Precision} \tag{4}$$

G-measure: 同 *F*1-score 一样, 它是综合考虑召回率和误报率, 来说明模型的检测性能. 计算公式如下:

$$G\text{-measure} = \frac{2\times Recall\times(1-pf)}{Recall+(1+pf)} \tag{5}$$

3.3 基线方法

传统跨项目缺陷预测问题与本文跨项目安全缺陷预测问题存在本质不同. 跨项目缺陷预测问题是根据软件历史开发数据以及已发现的缺陷, 借助机器学习等方法来预测出软件项目内的潜在缺陷程序模块, 主要关注点是程序内部模块, 是对于代码的分析. 而跨项目安全缺陷报告预测, 主要关注点是缺陷报告的文本数据, 是对缺陷报告的 Summary、Description 以及一些元数据的分析. 二者侧重点不同, 因此还是具有差别性的, 故本文没有将已有的跨项目缺陷预测方设为基线方法. 据我们所知只有 Peters 等人[9]与 Jiang 等人[14]在安全缺陷报告的跨项目预测领域有一定的研究, 并在高水平期刊发表相关论文. 为了验证知识图谱对于安全缺陷报告预测的性能表现, 我们使用他们的最新工作作为本文实验研究的基线. 二者工作都是从项目训练集缺陷报告的文本描述信息中提取安全关键词, 使用基于安全关键词的数据过滤方法, 并将其方法应用于项目内和跨项目安全缺陷报告检测中, 这两项工作与本文方法思路最为相似.

Peters 等人[9]在跨项目安全缺陷报告预测中所使用的方法为 FARSEC, 它是一个从 NSBR 中过滤噪声数据的框架. 它首先从训练数据的 SBR 中提取 top X (他们工作中使用的数字为 100) 安全相关关键字. 然后, 计算每个缺

陷报告 (在训练集中) 和这些安全关键字之间的相似性. 他们使用 7 种不同的过滤器从训练集中过滤掉噪声数据. 在此基础上, 我们采用基于规则的命名实体识别方法提取缺陷报告中实体与实体之间的关系, 并构建知识图来识别 SBR.

Jiang 等人[14]针对 FARSEC 方法在安全关键词选取不准确的问题, 提出了一种将关键词排名与词嵌入相结合的 LTRWES 方法. 该方法使用排名模型来有效过滤与 SBR 具有更高相似度的 NSBR. 其次, 它使用训练好的词嵌入模型将 NSBR 与 SBR 一起转化为低维实值向量, 实现安全缺陷报告更加准确的向量表示. 他们的结果表明 LTRWES 方法在对安全缺陷报告跨项目识别中具有更高的准确性.

3.4 检测方法及其流程

3.4.1 检测方法

在第 2 节中, 我们主要使用手动分类来标记实体, 用 CWE 和 CVE 对其进行扩展, 对缺陷报告进行了三元组实体规则建立. 然后将创建的语料库通过 spaCy 工具的字典来添加命名实体. 我们使用创建的语料库对测试集进行实体识别, 在获得相关单词后, 我们可以识别缺陷报告.

基于 Peters 等人[9]和 Wu 等人[10]的研究基础, 本文以安全关键字过滤为检测思路. 为了提高预测的准确率且对安全缺陷报告有一个直观的展示, 我们使用上文的数据集再加上安全漏洞领域知识来构建软件安全漏洞知识图谱. 我们在标记数据的过程中将五个不同数据集中的安全漏洞统一提取出来并构建成三元组规则实体, 并结合安全漏洞领域知识 CWE 和 CVE 危险漏洞类别丰富了三元组规则实体, 最后将完整的三元组规则实体通过 Neo4j 数据图构建成安全漏洞知识图谱. 在本文中我们将构建好的知识图谱作为一个新的安全漏洞库用来检测 SBR, 知识图谱安全漏洞库优势在于: 第一, 库由手动选取数据集中的每一个安全漏洞, 并结合了权威领域知识, 安全漏洞丰富且权威. 第二, 不仅考虑对实体进行识别, 同时还利用不同实体间的规则关系进行识别, 确保了 SBR 预测的准确性.

图 3 显示了跨项目 SBR 预测的过程. 在跨项目预测之前, 我们先进行了项目内预测, 根据实验结果对模型及其检测阈值等做了调整, 力求先让模型达到最好识别效果, 为跨项目做准备. 跨项目预测中, 输入数据集是一个完整的句子, 我们使用 spaCy 工具对输入的句子进行自然语言处理, 首先对每个句子进行分词断句, 再而标记句子中每个单词的词性, 然后对其进行词性还原. 我们在测试集中通过我们创建的 EntityRuler 规则的 JSON 文件筛选出单词类型为 SRW 和 VT 实体. 由于本文构建的是软件安全漏洞知识图谱, 范围相对固定, 但是安全实体的概念没有一个明确的定义, 我们根据 Gegick 等人[15]和 Pletea 等人[34]的研究结合本文构建的安全漏洞知识图谱, 建立了部分高频安全关键字再结合 CWE Top25 最危险漏洞类别 (表 3) 一起共同作为安全漏洞知识图谱核心实体 (表 12), 如图 3 所示, 在检测过程中测试集缺陷报告如果存在本文构建的核心实体, 我们就将其认定为 SBR.

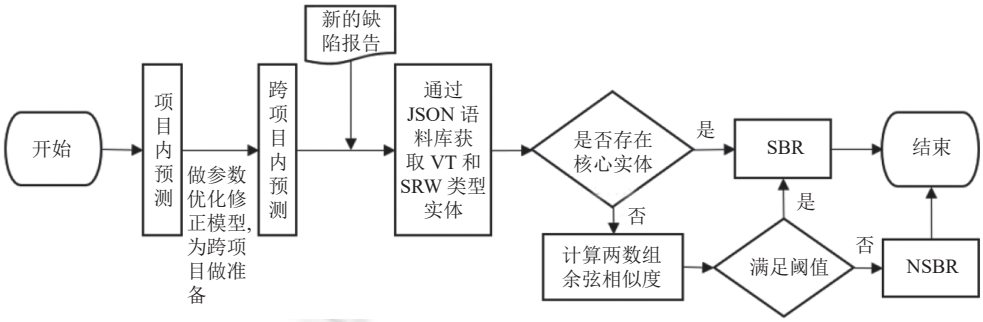

图 3 跨项目 SBR 预测流程

检测完毕后, 不存在核心实体的缺陷报告, 通过 JSON 文件匹配文本将其关键词筛选出存入新数组, 命名 Test. Chaparro 等人[28]通过对安全缺陷报告的实证研究, 表明了由报告者生成的软件缺陷报告, 其中包括对软件行为即观察到的行为 (OB)、步骤重现 (S2R) 和软件预期行为 (EB) 这 3 种内容描述为缺陷漏洞的主要描述形式, 且该描

述对开发人员在测试和修复缺陷时非常重要. 本文结合 Chaparro 等人的研究结果, 同时考虑安全缺陷报告描述特点 (即说明了安全漏洞产生原因、漏洞类型及其后果), 选取了与之对应的 3 种实体关系, 即 LT、CP 和 SP. 我们通过使用 Cypher 语句的 run 方法对 Test 数组中每一个实体, 获取所有与他相连的类型为 SRW 或 VT, 实体关系为 LT、CT、SP 的实体, 将满足上述条件的实体根据上文建立的规则, 并去除重复的实体后全部存入新建数组, 命名为 result, 然后计算 result 数组和和 Test 数组余弦相似度. 表 13 为我们在图谱中获取与单词“security”相连且满足上文条件的实体, 与 security 相连类型为 SRW 或者 VT 的满足关系的条件的实体共有 4 个.

表 12　核心实体及其个数

核心实体	核心实体个数
Denial of Service, sencryption, memory, exception, Execute Code, aurhorization, SSl, Overflow, npe, nullpointer, memory leak, Nullpointerexception, XSS Nullpointerexception	13

表 13　Security 连接的实体和关系

“n”	“m”	类型	关系
{“name”: “security”}	{“name”: “permission”}	SRW	LT
{“name”: “security”}	{“name”: “problem”}	SRW	CT
{“name”: “security”}	{“name”: “exception”}	SRW	LT
{“name”: “security”}	{“name”: “key password”}	SRW	LT
{“name”: “security”}	{“name”: “fail”}	SRW	PGS
{“name”: “security”}	{“name”: “break”}	SRW	PGS
{“name”: “security”}	{“name”: “security wizard”}	SRW	OR

在本文的工作中, 根据实际效果微调, 通过设置适当的阈值来确定它们是否相似, 根据实验效果我们将阈值设置为 0.5 得到最佳. 如果相似度大于 0.5, 则将其视为安全相关词. 对于本文方法来说, 余弦相似性有其优势性, 因为余弦相似度对角度敏感而维度不敏感, 即使两个相似的句子可能因其单词数量而相距很远 (例如, “bug”一词在一个句子中出现 8 次, 在另一个句子中出现 1 次), 但它们之间的角度可能仍然很小. 在第 4 节, 我们对本文方法进行了实验和评估.

3.4.2　实验流程

实验过程包括 3 个阶段. 首先, 我们为实体提取准备数据, 将每个数据集分成两个相等的部分, 第 1 部分生成语料库, 第 2 部分进行测试. 我们使用前 CWE 前 25 名最危险的漏洞类别和 CVE Details 作为本文工作的另一个来源. 之后, 我们使用 spaCy 标记数据集并恢复单词, 生成实体语料库并为每个数据集建立三元组实体关系规则. 将实体作为节点, 两个实体之间的关系作为 SBR 知识图的边. 最后, 将获得的所有三元组实体存储在 Neo4j 形成知识图谱中, 用于性能评估.

实验总体上我们做了两个方面的工作, WPP 和 TPP. 本文为了增强 TPP 的预测性能, 为 TPP 模型优化做准备. 先进行了 WPP, 根据实验结果调整模型参数. WPP 即使用项目中我们自己标记的数据标签, 来预测同一项目的未标记的缺陷报告中的 SBR. TPP 即使用来自一个项目中的标签来预测另外一个项目中未标记的缺陷报告中的 SBR.

在 WPP 中, 我们使用 5 个基本数据集, 每个数据集的前 50% 数据作为训练集, 结合 CWE Top25 最危险类别构建我们的知识图谱. 对新输入的句子, 我们对其进行安全相关词抽取和属性还原, 对于存在核心实体的缺陷报告, 我们鉴定其为安全缺陷报告, 不存在核心实体的缺陷报告在知识图谱中进行实体和关系相似度匹配, 超过我们设定的阈值就被认为是安全缺陷报告. 对于 TPP 我们使用 5 个数据集中的安全缺陷报告结合核心实体建立 5 个单知识图谱进行数据集之间的相互预测, 并使用 5 个数据集构建的一张大型知识图谱对两个大型数据集进行结果预测. 检测方法同 WPP. 在 5 个小型数据集的验证中, 我们将每个数据集作为目标项目, 其他 4 个数据集分别作为源项目进行预测. 在大型数据集的验证中, 我们将两个大型数据集作为目标项目, 5 个小型数据集作为源项目进行预测.

4 实验分析

4.1 针对 RQ1 的结果分析

RQ1: KG-SBRP 方法与已有基准方法相比是否可以提升跨项目安全缺陷报告预测性能?

TPP 在 7 个数据集上进行了实验, 其中前 5 个小型数据集我们通过构建 5 张单独的知识图谱进行交叉相互预测以此模拟跨项目预测实验结果如表 14 所示 (注: 粗体显示的是 *F*1-score 与 *G*-measure 最高的行). 结果显示 KG-SBRP 在 TPP 上的性能会高于 WPP 的性能, 可能的原因如下: 1) KG-SBRP 中改进了关键词抽取方法, 提高了关键词抽取质量; 并且, 在实体关系抽取中给予安全缺陷报告描述的通用知识更多关注, 从而降低不同项目之间差异特征影响. 2) 在执行 TPP 之前做了 WPP 强化训练, 提取出关键实体集合注入了不同数据集, 在很大程度上缓解了数据分布差异.

表 14 TPP 实验结果

目标数据集	源项目	召回率	误差率	精确率	*F*1-score	*G*-measure
Ambari	Ambari	0.81	0.01	0.62	0.13	0.89
	Camel	0.31	0.05	0.16	0.21	0.48
	Derby	0.88	0.32	0.08	0.15	0.76
	Wicket	0.25	0.18	0.04	0.07	0.38
	Chromium	**0.75**	**0.06**	**0.27**	**0.39**	**0.83**
Camel	Ambari	0.59	0.29	0.18	0.26	0.64
	Camel	0.59	0.30	0.16	0.26	0.64
	Derby	0.72	0.37	0.16	0.27	0.67
	Wicket	**0.67**	**0.31**	**0.18**	**0.28**	**0.42**
	Chromium	0.33	0.12	0.21	0.25	0.48
Derby	Ambari	0.75	0.47	0.28	0.40	0.62
	Camel	0.39	0.20	0.31	0.35	0.52
	Derby	0.86	0.56	0.27	0.41	0.59
	Wicket	**0.54**	**0.20**	**0.36**	**0.43**	**0.65**
	Chromium	0.33	0.14	0.35	0.34	0.48
Wicket	Ambari	0.65	0.29	0.10	0.17	0.68
	Camel	**0.39**	**0.09**	**0.17**	**0.24**	**0.55**
	Derby	0.83	0.27	0.12	0.22	0.78
	Wicket	0.87	0.43	0.09	0.16	0.69
	Chromium	0.43	0.18	0.10	0.16	0.56
Chromium	Ambari	0.44	0.27	0.04	0.07	0.55
	Camel	**0.43**	**0.03**	**0.25**	**0.31**	**0.65**
	Derby	0.86	0.22	0.08	0.15	0.82
	Wicket	0.49	0.07	0.14	0.22	0.64
	Chromium	0.68	0.17	0.08	0.15	0.75

在大型数据集上, Peters 等人[9]的 FARSEC 方法并没有实现大型数据集的 TPP 实验, 所以我们将本文复现的 FARSEC 用于大型数据的检测作为 KG-SBRP 方法的基线对比. 大型数据集知识图谱检测方面, 本文用 5 个小型数据集提取出的数据检测两个大型数据. 我们筛选出本实验与 FARSEC[9]在 *F*1-score 和 *G*-measure 上最好的实验结果作为对比. 结合表 14 中最好的数据, 和大型数据集的检测, 实验结果如表 15 所示.

表 15 中 7 个数据集每个对应两种方法的实验结果, 其中加粗行表示各数据在两种方法中 *F*1-score 与 *G*-measure 最高的实验结果, 可以看出在 7 个数据集上, 本文 KG-SBRP 方法在 *F*1-score 和 *G*-measure 上的结果基本高于基线 FARSEC 方法. 在 7 个数据集中, 本方法同 FARSEC 方法相比, *F*1-score 平均提高了 11%. *G*-measure 平均提高了 10%.

表 15 TPP 综合实验结果

目标数据集	方法	源项目	召回率	误差率	精确率	*F*1-score	*G*-measure
Ambari	KG-SBRP	Chromium	**0.75**	**0.06**	**0.27**	**0.39**	**0.89**
	FARSEC	Camel	0.14	0.20	0.50	0.22	0.59
Camel	KG-SBRP	Wicket	**0.67**	**0.31**	**0.18**	**0.28**	**0.67**
	FARSEC	Ambari	0.28	0.07	0.12	0.16	0.56
Derby	KG-SBRP	Wicket	**0.54**	**0.20**	**0.36**	**0.43**	**0.65**
	FARSEC	Ambari	0.19	0.02	0.47	0.27	0.58
Wicket	KG-SBRP	Camel	**0.39**	**0.09**	**0.17**	**0.24**	**0.78**
	FARSEC	Chromium	0.17	0.20	0.50	0.25	0.63
Chromium	KG-SBRP	Camel	**0.43**	**0.03**	**0.25**	**0.31**	**0.65**
	FARSEC	Ambari	0.46	0.20	0.12	0.18	0.64
Mozilla_Large	KG-SBRP	KG	**0.32**	**0.02**	**0.25**	**0.32**	**0.48**
	FARSEC	Ambari	0.23	0.08	0.12	0.21	0.46
Chromium_Large	KG-SBRP	KG	**0.25**	**0.02**	**0.19**	**0.29**	**0.40**
	FARSEC	Derby	0.55	0.24	0.18	0.18	0.38

与 KG-SBRP 方法不同, Jiang 等人[14]的 LTRWES 方法主要使用 *G*-measure 指标而非双指标来进行模型性能评估. 故我们选取 KG-SBRP 与 LTRWES 方法在 5 个数据集上实验结果最好 *G*-measure 值进行对比. 结果如表 16 所示.

表 16 KG-SBRP 方法与 LTRWES 的最优 *G*-measure 对比

目标数据集	方法	*G*-measure
Ambari	KG-SBRP	**0.89**
	LTRWES	0.72
Camel	KG-SBRP	**0.67**
	LTRWES	0.58
Derby	KG-SBRP	**0.65**
	LTRWES	0.61
Wicket	KG-SBRP	**0.78**
	LTRWES	0.65
Chromium	KG-SBRP	0.65
	LTRWES	**0.69**

表 16 中 5 个数据集对应两种方法的实验结果, 其中加粗行代表每个数据集在不同方法上 *G*-measure 的最高值. 由实验结果可以看出前 4 个数据集 *G*-measure 均高于 LTRWES 方法, 在 Chromium 数据集上 KG-SBRP 方法稍逊于 LTRWES 方法, 但二者差距很小. 总体来讲 KG-SBRP 方法在跨项目安全缺陷报告预测要稍微优于 LTRWES 方法.

为了从另一个方面验证 KG-SBRP 方法的有效性, 在同样的数据集项目内预测情况下我们使用了最新深度学习模型并结合该领域最新研究成果[36]和跨项目 KG-SBRP 方法进行有效性对比. 由于选取的方法使用 *F*1-score 作为主要判断指标, 故我们同样选取 KG-SBRP 方法最佳 *F*1-score 与其进行对比. 我们将 *F*1-score 高的行使用了加粗表示. 结果如后文表 17 所示.

项目内预测与跨项目预测结果比对必然存在差距. 但由后文表 17 可知, 不同方法在大多数数据集上差距并不是大, 有些数据集上还非常接近, 在 Wicket 与 Chromium 数据集上 KG-SBRP 方法甚至超过了深度学习方法. 深度学习方法在不同数据集上的实验有一定差距, 在稳定性方法与 KG-SBRP 方法存在一定差距.

4.2 针对 RQ2 的结果分析

RQ2: KG-SBRP 方法在跨项目安全缺陷报告预测性能上与传统深度学习方法相比有何优势?

本文做了以下 4 个方面工作来回答问题 RQ2.

(1) 提高了关键字的质量

本文以安全关键字过滤为检测思路, 因此单词标记的准确性也将直接影响本文方法的有效性和所构建图谱的质量. 为了避免了主观性造成的分类不准确, 屏蔽不同人员标记的差异, 我们对跨项目安全缺陷报告研究的常用 5 个数据集中所有安全缺陷报告进行了统一手工标记实体并建立实体规则. 并且将 VT (漏洞类型) 和 SRW (安全相关词) 类型为我们检索着重要关注的实体, 同时结合 CWE 和 CVE 扩充了我们的实体语料库, 这极大地提高了安全关键字的质量. Peters 等人[9]通过 TF-IDF 方法提取数据集描述文本中的评分最高的 100 个单词作为安全关键词词, 经过 Zheng 等人[37,38] 通过实证研究发现, Peter 等人的方法抽取的安全关键字质量较差, 抽取的安全关键词大多似乎与安全无关, 导致检测效果不佳. 如表 18 所示, FARSEC 方法中提取的 Chromium 数据集中的 100 个安全关键词 (粗体为与安全似乎无关的词), 从结果可以看出其中大多数可能与安全无关, 有些单词甚至可能出现在每个缺陷报告中 (例如: “starred”“notified”“may”等).

表 17 KG-SBRP 方法与深度学习方法的最优 *F*1-score 对比

目标数据集	方法	*F*1-score
Ambari	KG-SBRP	0.39
	Attention+TextCNN	**0.61**
Camel	KG-SBRP	0.28
	TextCNN	**0.46**
Derby	KG-SBRP	0.43
	Attention+TextRNN	**0.46**
Wicket	KG-SBRP	0.24
	TextCNN	**0.18**
Chromium	KG-SBRP	0.31
	TextRNN	**0.29**

表 18 Chromium 数据集提取的安全相关词及其个数

关键词	个数
file, security, **chrome, page, http**, download, **user**, starred, person, notified, changes, **may**, **see**, **url**, **site**, **bug**, **open**, **google**, **browser**, **like**, **windows, window**, https, web, code, **one**, memory, firefox, **function, tests**, problem, seems, **tab, also**, version, **use, would**, **using**, **view**, **used**, **make**, **users**, **chromium**, crash, **click**, password, **think**, vulnerability, **sure**, **browsers**, **link**, attached, attacker, data, **get**, **fix**, const, **content**, **something**, **safari**, **new**, error, javascript, **lcamtuf**, malicious, **please**, **could**, risk, release, **try**, **found**, allow, expected, **time**, **example**, corruption, **test**, back, access, crashes, **urls**, **int**, **without**, **know**, **versions**, **way**, uses, order, report, cause, fail, **want**, **system**, **still**, **files**, arbitrary, **html**, details, ssl, **need**, loaded, **might**	100

(2) 经过两次训练提升 KG-SBRP 方法的检测精度

在跨项目预测上, 本文先进行了本项目训练, 即 WPP. WPP 是在对 TPP 做一些参数优化调整, 使 TPP 能够达到更好的效果, 以此为跨项目识别做准备对跨项目进行强化学习. 这方面我们主要做了 3 个方面的工作. 1) 修正标记实体及其规则的准确性. 使用 5 个数据集的三元组规则建立了一张完整的软件安全漏洞知识图谱, 使用这张知识图谱对 5 个数据集进行安全缺陷报告预测, 根据实验的检测结果输出混淆矩阵, 我们根据混淆矩阵的结果, 不断修正调节标记的实体及其实体间的关系. 2) 确定合适的检测阈值. 在报告不存在核心实体的情况下, 为求在每一个数据集的知识图谱检测都能达到最好效果, 我们通过二分检索法确定两个数组之间合适的余弦阈值, 结合 Zheng 等人[38]的实证研究, 最终确定合适的检测阈值. 3) 缓解不同项目数据集差异. 使用 WPP 构建的知识图谱从中提取出关键实体集合核心实体注入 5 个数据集构建的 5 张知识图谱, 以此增加不同数据集之间的共性, 很大程度上缓解不同项目之间的数据分析差异. 而在 FARSEC 方法中, Peters 等人[9]通过筛选过滤掉不同数据源中的噪音 NSBR, 未考虑 SBR 和 NSBR 中的联系且不同项目之间得数据分布差异仍然存在.

(3) 安全缺陷报告判断标准更加严格

当某些缺陷报告不存在核心实体时, 通过 JSON 文件抽取到的实体, 需要根据获取该实体在知识图谱中搜索连通的安全相关类型单词, 根据这些安全相关词汇进行规则识别匹配. 如果句子中安全相关词汇及其之间规则检测都满足要求才将其视为安全相关缺陷报告. 这里本文设置了双层要求来鉴别 SBR, 在一定程度上减少了噪音的影响. 而 Perters 等人[9]和 Wu 等人[10]的方法仅考虑缺陷报告中单个安全关键词来判断安全缺陷报告, 导致了预测结果不理想.

例如 Chromium 数据集中编号为 59 的缺陷报告描述: “If I let Chrome remember my passwords they are available to anyone who has access to my computer (including thieves). I would like to protect my saved passwords with a master password to circumvent this security issue.”, 该句中不存在核心实体, 但存在关键词 security 我们可以通过 JSON 文件检测到. 因此还需要在知识图谱中进行实体和规则关系匹配再次检测.

如表 19 所示, 使用命令 Match(n:ad{name:“security”})–(m:ad) 返回 n, m, 可以在知识图谱中查找到与 security 实体相连的所有安全相关单词和短语以及他们之间的规则关系. 可以发现与安全实体相关的实体有权限、问题、异常、密钥密码、失败、中断、安全向导等. 结合安全漏洞报告特性, 我们仅提取与 security 连接实体类型为 SRW 或 VT 且关系为 LT、CT、SP 之一的实体, 并进行余弦值计算. 结果显示该报告未达到我们设定的阈值, 所以不是安全缺陷报告, 判断结果与实际一致. 如果使用 Peters 等人[9]的 FARSEC 方法, 该缺陷报告中存在表 18 中列出的多个安全关键字 (Chrome、password 等), 通过计算该报告将被认定为 SBR, 导致预测错误.

表 19 Security 连接的实体和关系

“n”	“m”	类型	关系
{“name”: “security”}	{“name”: “permission”}	SRW	LT
{“name”: “security”}	{“name”: “problem”}	SRW	CT
{“name”: “security”}	{“name”: “exception”}	SRW	LT
{“name”: “security”}	{“name”: “key password”}	SRW	LT
{“name”: “security”}	{“name”: “fail”}	SRW	PGS
{“name”: “security”}	{“name”: “break”}	SRW	PGS
{“name”: “security”}	{“name”: “security wizard”}	SRW	OR

(4) 考虑关键词之间的语义信息, 增添了短语验证

在检测的过程中, 本文还增添了短语验证. 因为有一些重要关键词是以短语结合进行出现. 例如在短语验证错误中, 单词 validation 与安全性无关, 但 validation error 就被视为漏洞类型需要标记. 还有 Command 和 Injection 分开来看毫无关系, Command 意为命令控制等, Injection 意为注射注入等, 但是在安全漏洞领域内合在一起 Command-Injection 就是一种极为关键的漏洞类型, 是 CWE Top25 之一. 本文在手工标记安全缺陷报告过程中, 根据其特征, 总结了数据集中的关键短句写入 JSON 文件中. 例如 Chrome 数据集中编号为 1 的安全缺陷报告描述: “Init Wizard: Advanced config validation errors can be bypassed”, 使用基线 FARSEC 方法进行检测, 该句中不存在任何表 18 中的 100 个关键字, 所以会被错误的预测为 NSBR. 使用基线 Keyword matrix 方法检测时, 该句中也不存在 CWE 和 CVE 的词汇, 所以也会被错误的认定为 NSBR. 但使用本文 KG-SBRP 方法, validation error 短语通过手工标记的时候已经加入 JSON 检测文件汇入到图谱中, 所以该报告会被预测为安全缺陷报告.

5 讨 论

5.1 KG-SBRP 方法的优势

相比传统数据存储和计算方式, KG-SBRP 方法以知识图谱为基础进行检测, 其优势体现在以下 3 个方面.

(1) 知识图谱将互联网上海量庞大的内容信息转化为机器可以理解和计算的知识形式. 它关系的表达能力比较强, 将数据通过图论模型的表示出来, 可以处理复杂多样的数据关联分析. 关系的层级及表达方式多种多样, 传

统数据通常只是通过表格、字段等方式进行展示, 难免造成部分数据或关系的遗漏. 从本文所建知识图谱中, 不同实体显示不同颜色, 实体关系也各不相同, 我们可以更好的理解安全实体之间的关系和特征, 降低了理解难度, 带来更好的视觉体验. 同时满足各种实体关系的分析和管理需要.

(2) 知识图谱自带语义、实体类型区别、和实体间的规则. 图谱中的每一个结点对应我们现实世界中的一个实体或者概念, 实体间相连的边代表着实体之间的关系. 在此之上, 我们还可以根据自己定义的规则和实体间的关系, 推导出知识图谱中没有明确表达出的知识. 比如, 知识图谱中有一对实体: security 同 permission, 他们之间的关系为 LT (起因, 造成), 可以推测在权限申请上可能会造成安全问题, 同 permission 相连的实体还有 password fail, 这样还推断出在密码失败申请上会出现安全问题, 然而 fail 又相连了实体 ssl, 这样又能推断出一些重要信息. 当这条线越来越长越多的时候, 我们就可以更加深入挖掘出这个错误造成的原因和错误出现的源头等更多的重要信息, 帮助我们更好地去理解安全报告. 线越来越多汇聚成图的时候, 它不需要中间过程的转换和处理就能表达安全缺陷报告各个实体的各种关系, 并且直观、自然、高效.

(3) KG-SBRP 方法可以用在非安全类型缺陷报告预测中, 实际上缺陷报告在分为安全缺陷报告或非安全缺陷报告可视为一个二元性分类问题, 本文方法已经解决了安全缺陷报告的预测的问题, 而该问题的反面就是非安全缺陷报告. 当不满足本文的安全缺陷报告识别任一条件, 该报告就是非安全类型的报告. 本文知识图谱方法从这样的观点来看, 也可以识别非安全缺陷报告.

5.2 知识图谱的构建成本

我们基于 Peters 等人[9]和 Wu 等人[10]清理共享的 5 个数据集, 即 Ambari、Camel、Derby、Wicket、Chromium 构建本文领域知识图谱. 知识图谱的构建开销主要分为两个方面: 第 1 个方面是规则建立方面. 本文是首次使用知识图谱结合软件安全漏洞领域知识来改进跨项目 SBR 预测性能的团队, 故需要查询相关资料, 并手工检查并定义实体之间的关系. 本文基于安全关键字思路预测识别安全缺陷报告, 通过卡片分类法对 5 个数据集中的所有安全缺陷报告进行了实体标记, 建立实体关系, 最终构建三元组规则实体. 在这一点上需要人力和时间资源的开销. 第 2 个方面是知识图谱的建立方面. Neo4j 数据库提供了良好的数据接口, 我们可以通过官方提供的教程编写代码直接连接数据库, 将我们的三元组规则实体在 Neo4j 数据库中进行图形可视化. 这一点上构建图谱时间资源开销可以完全忽略不计.

5.3 知识图谱的维护成本

我们构建一张通用软件安全漏洞知识图谱, 为该领域的研究提供了参考. 故当有新项目需要进行跨项目安全缺陷报告预测时, 并不需要重新构建知识图谱, 可以直接参考或使用本文构建的通用图谱来进行预测, 本文的实验结果证明了该图谱的有效性. 提取出 5 个数据集安全缺陷报告的特征及其内部联系, 相当于构建了一个安全缺陷报告关键词的“部分标准答案”, 当有新数据集需要预测时, 可以直接与我们构建的“部分标准答案”比对进行预测. 未来, 随着科学技术的发展, 软件规模会逐渐扩大, 软件安全漏洞的数量也会不断上升, 越来越多的命名实体在肯定会不断地出现, 例如新的软件名和新的缺陷名字. 这将在一定程度上影响本文预测方法的准确性. 因此, 还需要从以下几点进行改进, 提升知识图谱的质量. 1) 引入更多缺陷报告特征. 本文目前仅考虑到缺陷报告的文本信息(例如, 摘要和描述等), 并未采用缺陷报告的其他特征信息. 缺陷报告除去文本信息外还存在大量元数据, 例如: 模块、优先级、重要性等. 可以通过引入更多的缺陷报告特征信息来丰富领域知识提升安全缺陷报告检测的准确性. 同时目前对元数据的选取没有一个统一的标准, 元数据字段对于安全缺陷报告检测工作的性能提升有多大, 哪些元数据字段的检测可以明显改善安全缺陷报告的检测方法精度都是未来进一步研究的工作. 2) 缺陷报告数据集的优化. 本文目前所采用的数据集是由 Peters 等人[9]和 Wu 等人[10]贡献的安全缺陷报告检测常用共有数集, 但在公有数据集上, 很多缺陷报告的描述信息上存在一定缺陷. 如 Camel 数据集中编号为 13 的缺陷报告的描述: “Customer-serviceswsaddressingtesthangssometimes”, 单词之间衔接在一起, 导致部分单词无法识别出来, 这必定会对我们的阅读和模型检测结果造成一定影响. 后来的研究如果能够解决此类问题, 势必会再次提升模型的检测指标. 3) 针对安全缺陷报告类别数据扩展, 目前仅使用 Peters 等人[9]和 Wu 等人[10]所贡献的数据集尚不能完全满足

算法训练、学习的需要, 本文提出的 KG-SBRP 方法存在一定局限性, 当安全缺陷报告检测出的实体存在于图谱中才能够满足于本方法检测. 仅靠少量公有数据集所构建的安全漏洞知识图谱存在一定的知识缺陷和知识漏洞, 领域知识的质量也存在不足. 在未来, 希望能够引入更多的数据集来丰富扩展软件安全漏洞知识图谱, 提升安全缺陷报告的检测质量, 同时在更多数据集上对方法进行验证.

5.4 跨项目缺陷报告预测与知识图谱的关系

跨项目缺陷报告预测并不一定需要完全依赖于知识图谱, 知识图谱只是在一定程度上提升了安全缺陷报告预测准确性, 当缺陷报告预测符合某种合适条件时才会用到知识图谱. 当输入一个报告, 其中存在核心实体时, 该报告直接被预测为 SBR, 不存在核心实但存在安全关键词时, 此时才需要利用知识图谱选取对应实体关系进行 SBR 预测. 安全缺陷报告相对规范, 在跨项目缺陷报告预测上, 根据安全关键字过滤思路已有模型很难再提升预测的准确性, 知识图谱在考虑关键字过滤基础上考虑到了 SBR 内部联系, 增添了关键字语义信息以及实体关系验证等, 提升了预测准确性. 未来在知识图谱预测基础上引入更多缺陷报告特征与优化缺陷报告数据集等, 可以再次提升预测准确率. 从这一点上看, 在一定程度上跨项目安全缺陷报告预测对知识图谱更加依赖了.

5.5 有效性影响因素分析

● 内部有效性. 对本研究内部的有效性的存在以下 4 个威胁: 第 1 个威胁是该方法的实现正确性. 我们的研究基于 Peters 等人[9]和 Wu 等人[10]. 在本次实验中, 仔细检查并验证代码减少人为因素影响, 以确保其正确性. 第 2 个威胁是提取安全领域知识的来源. 为了确保我们获得的安全领域知识的质量, 我们使用 CWE 中软件漏洞的权威定义来提取常见的安全领域关键字. 此外, 我们使用历史 CVE 条目 (生产项目的基本安全问题) 作为数据源来提取项目特定的安全领域知识. 第 3 个威胁是安全相关单词的数量和表述准确性, 工作人员安全领域知识储备不同, 主观认知不同, 同一异常可能以不同的方式写入, 例如在对本文数据集进行标记时, 我们发现不同数据集“空指针异常有 3 种表方式”分别为: nullpointer、nullpionterexception 和 npe, 如果不加统一, 首先它们会以 3 种类型实体存入图谱, 随着同样数据的增加, 容易出现数据爆炸, 影响实验检测效率. 其次, 这 3 种类型会输入图谱后分别连接不同类型实体, JSON 文件检索和匹配实体关系时有一些安全关键实体就会检测不出来, 导致预测没有达到阈值, 就有可能影响实验结果. 同时同一问题以多种方式表达, 也可能会增加短语验证数量, 导致垃圾代码产生. 第 4 个威胁是安全缺陷类型的影响, 目前在跨项目安全缺陷报告预测领域中大多使用 Peter 等人[9]和 Wu 等人[10]清理和共享的 5 个数据集, 该数据集中的存在安全漏洞类型与实体关系等数据是有限的. 未来, 随着科学技术的发展, 软件规模会逐渐扩大, 软件安全漏洞的数量也会不断上升, 越来越多的命名实体在肯定会不断地出现, 例如新的软件名和新的缺陷名字. 这将在一定程度上影响所有使用这些数据集构建预测方法的准确性. 在研究中, 我们手动识别和标记每一组数据的安全相关词. 对于测试集中的数据, 我们通过计算余弦相似度来判断它是否是安全相关词汇. 我们建立关系生成知识图, 并通过搜索知识图谱获得安全相关词. 在某处的单词越多, 连接边越多, 它是安全缺陷报告的可能性越高. 如果检测报告中抽取的实体出现在知识图谱中多数实体汇聚的地方, 则该缺陷报告很大概率是 SBR.

● 外部有效性. 外部有效性主要涉及本文实验结果是否具有普遍性. 在实验中本文使用 7 个不同领域项目的公开数据集. 此外, 这些数据集具有不同的规模 (Chromium 和其他两个大型数据集有超过 40k 条记录, 而其他 4 个数据集各有 1 000 条记录). 所有这些都可以保证实验数据集的多样性. 因此, 本文使用的方法具有可靠性、通用性. 尽管如此, 这并不是说本文的方法和实验结果可以推广到所有软件信息检测任务. 然而, 它将适用于基于监督机器学习的 SBR 预测任务.

● 结论有效性. 结论有效性主要是指在本文实验结果中使用的检测评价指标是否合理. 本文使用了机器学习研究领域经常使用的几种指标, 同时同本文的基线数据来源 Peters 等人[9]和 Wu 等人[10]使用相同的指标, 即召回率、误差率、准确率、$F1$-score 和 G-measure 因此可以从多个角度对融合知识图谱的安全缺陷报告预测模型性能做出客观的评价.

6 总结与展望

SBR 预测是降低软件产品风险的重要途径, 本文提出了融合知识图谱的跨项目 SBR 预测 KG-SBRP 方法, 主要使用本文构建的软件安全漏洞知识图谱对缺陷报告进行 SBR 预测, 首先是对本文所用数据集的所有安全缺陷报告进行了统一手工选取实体标记实体属性, 以及实体之间的关系. 利用构建好的三元组规则实体构建本文的软件安全漏洞知识图谱. 然后在不同规模数据集进行项目内的训练和测试, 根据实验的检测结果对知识图谱调整, 为求在每一个数据集检测上达到最好效果. 以此为跨项目识别做准备. 最后在 7 个数据集上, 使用 KG-SBRP 方法进行了跨项目预测实验, 结果表明, KG-SBRP 方法的预测性能要显著优于当前流行的方法, 其性能指标 *F*1-score 在 TPP 平均提升 11%, *G*-measure 在 TPP 上提升 10%.

本文属于知识图谱在 SBR 预测研究中的初步探索, 还存在一定的缺陷和不足. 首先描述文本语句结构复杂, 手工定制规则工作量巨大. 不同的语言和不同的领域之间存在差异, 并且很难迁移所有的定制数据. 且不同工作人员对安全领域知识储备不同, 认知不同, 同一实体可能出现不同标记. 命名实体的类型多样, 新的命名实体在未来也会不断出现, 例如新的软件名称和缺陷类型. 此外, 命名实体的组成更为复杂, 同一种类型异常可能以不同的方式写入, 例如: nullpointer、nullpionterexception、npe. 这使得获取的安全相关词汇不准确. 在不同的字段和场景中, 命名实体的扩展是不同的, 这可能会导致歧义. 未来, 我们将通过引入更多缺陷报告的特征 (例如, 报告者、模型) 来丰富领域知识. 此外, 还可以合并一个知识图谱来更好地表示缺陷报告特征之间的关系. 知识图谱的使用提高了 SBR 预测的准确性. 因此, 我们将继续改进我们的知识图谱, 提高安全词汇的准确性, 以便更好地预测.

References:

[1] Amoroso E. Recent progress in software security. IEEE Software, 2018, 35(2): 11–13. [doi: 10.1109/MS.2018.1661316]

[2] CVE Website. 2021. https://www.openstack.org/

[3] CVE Detail. 2021. https://www.cvedetails.com/vendor/11727/Openstack.html

[4] Ambari. 2019. http://ambari.apache.org/

[5] Camel. 2019. http://camel.apache.org/

[6] Derby. 2019. http://db.apache.org/derby/

[7] Wicket. 2019. http://wicket.apache.org/

[8] Apache. 2019. http://db.apache.org/

[9] Peters F, Tun TT, Yu YJ, Nuseibeh B. Text filtering and ranking for security bug report prediction. IEEE Trans. on Software Engineering, 2019, 45(6): 615–631. [doi: 10.1109/TSE.2017.2787653]

[10] Wu XX, Zheng W, Xia X, Lo D. Data quality matters: A case study on data label correctness for security bug report prediction. IEEE Trans. on Software Engineering, 2022, 48(7): 2541–2556. [doi: 10.1109/TSE.2021.3063727]

[11] Behl D, Handa S, Arora A. A bug mining tool to identify and analyze security bugs using naive Bayes and TF-IDF. In: Proc. of the 2014 Int'l Conf. on Reliability Optimization and Information Technology. Faridabad: IEEE, 2014. 294–299. [doi: 10.1109/ICROIT.2014.6798341]

[12] Shu R, Xia TP, Williams L, Menzies T. Better security bug report classification via hyperparameter optimization. arXiv:1905.06872, 2019.

[13] Goseva-Popstojanova K, Tyo J. Identification of security related bug reports via text mining using supervised and unsupervised classification. In: Proc. of the 2018 IEEE Int'l Conf. on Software Quality, Reliability and Security (QRS). Lisbon: IEEE, 2018. 344–355. [doi: 10.1109/QRS.2018.00047]

[14] Jiang Y, Lu PC, Su XH, Wang TT. LTRWES: A new framework for security bug report detection. Information and Software Technology, 2020, 124: 106314. [doi: 10.1016/j.infsof.2020.106314]

[15] Gegick M, Rotella P, Xie T. Identifying security bug reports via text mining: An industrial case study. In: Proc. of the 7th IEEE Working Conf. on Mining Software Repositories. Cape Town: IEEE, 2010. 11–20. [doi: 10.1109/MSR.2010.5463340]

[16] Wijayasekara D, Manic M, Wright JL, McQueen M. Mining bug databases for unidentified software vulnerabilities. In: Proc. of the 5th Int'l Conf. on Human System Interactions. Perth: IEEE, 2012. 89–96. [doi: 10.1109/HSI.2012.22]

[17] Camargo Cruz AE, Ochimizu K. Towards logistic regression models for predicting fault-prone code across software projects. In: Proc. of the 3rd Int'l Symp. on Empirical Software Engineering and Measurement. Lake Buena Vista: IEEE, 2009. 460–463. [doi: 10.1109/ESEM.

2009.5316002]

[18] Peters F, Menzies T, Marcus A. Better cross company defect prediction. In: Proc. of the 10th Working Conf. on Mining Software Repositories. San Francisco: IEEE, 2013. 409–418. [doi: 10.1109/MSR.2013.6624057]

[19] Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B. Cross-project defect prediction: A large scale experiment on data vs. domain vs. process. In: Proc. of the 7th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering. Amsterdam: ACM, 2009. 91–100. [doi: 10.1145/1595696.1595713]

[20] Nam J, Pan SJ, Kim S. Transfer defect learning. In: Proc. of the 35th Int'l Conf. on Software Engineering (ICSE). San Francisco: IEEE, 2013. 382–391. [doi: 10.1109/ICSE.2013.6606584]

[21] Turhan B, Menzies T, Bener AB, Di Stefano J. On the relative value of cross-company and within-company data for defect prediction. Empirical Software Engineering, 2009, 14(5): 540–578. [doi: 10.1007/s10664-008-9103-7]

[22] Wu F, Jing XY, Sun Y, Sun J, Huang L, Cui FY, Sun YF. Cross-project and within-project semisupervised software defect prediction: A unified approach. IEEE Trans. on Reliability, 2018, 67(2): 581–597. [doi: 10.1109/TR.2018.2804922]

[23] Zhong S, Khoshgoftaar TM, Seliya N. Unsupervised learning for expert-based software quality estimation. In: Proc. of the 8th IEEE Int'l Symp. on High Assurance Systems Engineering. Tampa: IEEE, 2004. 149–155. [doi: 10.1109/HASE.2004.1281739]

[24] Xia X, Lo D, Qiu WW, Wang XG, Zhou B. Automated configuration bug report prediction using text mining. In: Proc. of the 38th IEEE Annual Computer Software & Applications Conf. Vasteras: IEEE, 2014. 107–116. [doi: 10.1109/COMPSAC.2014.17]

[25] 2021 Top25 CWEs. 2021. https://cwe.mitre.org/top25/

[26] Martin B, Brown M, Paller A, Kirby D, Christey S. 2011 CWE/SANS top 25 most dangerous software errors. 2011. https://cwe.mitre.org/top25/archive/2010/2010_cwe_sans_top25.pdf

[27] Chawla I, Singh SK. Automatic bug labeling using semantic information from LSI. In: Proc. of the 7th Int'l Conf. on Contemporary Computing (IC3). Noida: IEEE, 2014. 376–381. [doi: 10.1109/IC3.2014.6897203]

[28] Chaparro O, Lu J, Zampetti F, Moreno L, Di Penta M, Marcus A, Bavota G, Ng V. Detecting missing information in bug descriptions. In: Proc. of the 11th Joint Meeting on Foundations of Software Engineering. Paderborn: ACM, 2017. 396–407. [doi: 10.1145/3106237.3106285]

[29] Lu PC. Security bug report identification and bug localization based on deep learning [MS. Thesis]. Harbin: Harbin Institute of Technology, 2019 (in Chinese with English abstract). [doi: 10.27061/d.cnki.ghgdu.2019.001248]

[30] Yang XL, Lo D, Huang Q, Xia X, Sun JL. Automated identification of high impact bug reports leveraging imbalanced learning strategies. In: Proc. of the 40th IEEE Annual Computer Software and Applications Conf. (COMPSAC). Atlanta: IEEE, 2016. 227–232. [doi: 10.1109/COMPSAC.2016.67]

[31] Zhou YQ, Sharma A. Automated identification of security issues from commit messages and bug reports. In: Proc. of the 11th Joint Meeting on Foundations of Software Engineering. Paderborn: ACM, 2017. 914–919. [doi: 10.1145/3106237.3117771]

[32] Yang B, Xing ZC, Xia X, Chen CY, Ye DH, Li SP. Don't do that! Hunting down visual design smells in complex UIS against design guidelines. In: Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Madrid: IEEE, 2021. 761–772. [doi: 10.1109/ICSE43902.2021.00075]

[33] Fan YR, Xia X, Lo D, Hassan AE. Chaff from the wheat: Characterizing and determining valid bug reports. IEEE Trans. on Software Engineering, 2020, 46(5): 495–525. [doi: 10.1109/TSE.2018.2864217]

[34] Pletea D, Vasilescu B, Serebrenik A. Security and emotion: Sentiment analysis of security discussions on GitHub. In: Proc. of the 11th Working Conf. on Mining Software Repositories. Hyderabad: ACM, 2014. 348–351. [doi: 10.1145/2597073.2597117]

[35] Xia X, Lo D, Shihab E, Wang XY, Zhou B. Automatic, high accuracy prediction of reopened bugs. Automated Software Engineering, 2015, 22(1): 75–109. [doi: 10.1007/s10515-014-0162-2]

[36] Zheng W, Chen JZ, Wu XX, Chen X, Xia X. Empirical studies on deep-learning-based security bug report prediction methods. Ruan Jian Xue Bao/Journal of Software, 2020, 31(5): 1294−1313 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5954.htm [doi: 10.13328/j.cnki.jos.005954]

[37] Zheng W, Chen Z, Wu XX, Fu WQ, Sun BW, Cheng JY. A domain knowledge-guided lightweight approach for security bug reports prediction. In: Proc. of the 8th Int'l Conf. on Dependable Systems and Their Applications (DSA). Yinchuan: IEEE, 2021. 359–368. [doi: 10.1109/DSA52907.2021.00055]

[38] Zheng W, Cheng JY, Wu XX, Sun RY, Wang XL, Sun XB. Domain knowledge-based security bug reports prediction. Knowledge-based Systems, 2022, 241: 108293. [doi: 10.1016/j.knosys.2022.108293]

附中文参考文献:

[29] 路鹏程. 基于深度学习的安全缺陷报告识别和缺陷定位 [硕士学位论文]. 哈尔滨: 哈尔滨工业大学, 2019. [doi: 10.27061/d.cnki.ghgdu.2019.001248]

[36] 郑炜, 陈军正, 吴潇雪, 陈翔, 夏鑫. 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. http://www.jos.org.cn/1000-9825/5954.htm [doi: 10.13328/j.cnki.jos.005954]

郑炜(1975－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为软件测试, 软件安全, 软件仓库挖掘.

成婧源(1998－), 女, 硕士生, 主要研究领域为知识图谱, 软件安全.

刘程远(1994－), 男, 硕士, CCF 学生会员, 主要研究领域为知识图谱, 安全缺陷报告识别, 深度学习.

孙小兵(1985－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为智能化软件数据分析, 软件安全.

吴潇雪(1983－), 女, 博士, 助理教授, 主要研究领域为软件测试, 智能化软件漏洞分析.

孙瑞阳(1994－), 男, 硕士, 主要研究领域为软件缺陷预测.

陈翔(1980－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为软件缺陷预测, 软件缺陷定位, 回归测试, 组合测试.