计算机科学
COMPUTER SCIENCE

结合开发者依赖的图神经网络缺陷预测方法

乔羽, 徐涛, 张亚, 文凤鹏, 李强伟

引用本文

乔羽, 徐涛, 张亚, 文凤鹏, 李强伟. 结合开发者依赖的图神经网络缺陷预测方法[J]. 计算机科学, 2025, 52(6): 52-57.

QIAO Yu, XU Tao, ZHANG Ya, WEN Fengpeng, LI Qiangwei. Graph Neural Network Defect Prediction Method Combined with Developer Dependencies [J]. Computer Science, 2025, 52(6): 52-57.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240700119

结合开发者依赖的图神经网络缺陷预测方法

乔　羽[1]　徐　涛[2]　张　亚[1]　文凤鹏[1]　李强伟[1]
1 南京航空航天大学计算机科学与技术学院　南京 211106
2 枣庄学院网络中心　山东 枣庄 277015
(cklqiaoyu@126.com)

摘　要　在软件开发过程中,及时识别和处理高风险缺陷模块是至关重要的。传统的软件缺陷预测方法主要基于代码相关的信息,但常常忽略了开发者个人特质对软件质量的影响。针对这一问题,提出了一种新型的结合开发者一致性依赖网络的软件缺陷预测模型DCN4SDP。首先利用开发者信息构建了一个开发者一致性依赖网络,并提取代码相关的度量作为网络的初始度量元,通过使用双向门控图神经网络学习网络结构上的节点特征。实验结果表明,DCN4SDP模型在多个标准数据集上的性能显著优于传统机器学习分类器和其他深度学习方法,AUC值达到了0.91,F1值达到了0.76,均显著高于其他对比模型。这些优势表明将开发者维度融入软件缺陷预测能够有效提升模型的预测能力和应用价值,且为未来的软件缺陷预测研究提供了新的思路和方向。

关键词:软件缺陷预测;双向门控图神经网络;开发者信息;深度学习;图神经网络;软件工程
中图分类号　TP311

Graph Neural Network Defect Prediction Method Combined with Developer Dependencies

QIAO Yu[1],XU Tao[2],ZHANG Ya[1],WEN Fengpeng[1] and LI Qiangwei[1]
1 Department of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China
2 Network Center,Zaozhuang University,Zaozhuang,Shandong 277015,China

Abstract　In the software development process,timely identification and handling of high-risk defect modules are crucial. Traditional software defect prediction methods primarily rely on code-related information but often overlook the impact of developers' personal characteristics on software quality. To address this issue,this study proposes a novel software defect prediction model,DCN4SDP,which incorporates a developer consistency dependency network. This model first constructs a developer consistency dependency network using developer information and extracts code-related metrics as initial features for the network. It then employs a bidirectional gated graph neural network(BiGGNN) to learn the node features within the network structure. Experimental results demonstrate that the DCN4SDP model significantly outperforms traditional machine learning classifiers and other deep learning methods on multiple standard datasets. For instance,the DCN4SDP achieves an AUC value of 0.91 and a F1 score of 0.76,both notably higher than those of other compared models. These advantages indicate that integrating the developer dimension into software defect prediction can effectively enhance the model's predictive capabilities and practical value,providing new insights and directions for future research in software defect prediction.

Keywords　Software defect prediction,Bidirectional gated graph neural network,Developer information,Deep learning,Graph neural network,Software engineering

1　引言

在软件开发过程中,不可避免地会产生软件缺陷[1-2],因此及时识别高风险软件缺陷模块,进而有效分配资源(如预算、时间和人员)对开发人员至关重要[2-3]。目前,开发人员和研究者常通过统计计算或机器学习算法学习软件内部属性(如代码度量元、过程度量元)进行软件缺陷预测(Software Defect Prediction,SDP),帮助开发者自动快速地发现高风险缺陷模块[1]。

到稿日期:2024-07-18　返修日期:2024-09-02
基金项目:国家自然科学基金(62202223);江苏省自然科学基金(BK20220881);工信部安全关键软件重点实验室(南京航空航天大学)开放项目(NJ2022027)
This work was supported by the National Natural Science Foundation of China(62202223),Natural Science Foundation of Jiangsu Province,China(BK20220881),Open Project of the Key Laboratory of Security Critical Software of the Ministry of Industry and Information Technology(Nanjing University of Aeronautics and Astronautics)(NJ2022027).
通信作者:徐涛(xutao@uzz.edu.cn)

先前的研究已经证实依赖网络度量元可以提高 SDP 模型的性能[4-5]。目前大多数研究使用软件模块的代码依赖关系进行缺陷预测，如代码之间的数据依赖或者调用依赖[4-5]。然而，仅从代码角度提取的网络结构忽略了复杂的、多人员协作的软件开发过程中其他角度（如开发者角度）的信息，限制了通过这些度量生成的模型覆盖各种缺陷的能力。据我们所知，开发过程中产生缺陷的原因离不开开发者。开发者会在不同模块中表现出不同的编码风格、提交频率和经验水平，导致产生不同的缺陷模式[6]。例如，由同一开发者开发的软件模块如果存在不良的编码习惯，则会重复出现相同类型的代码缺陷，这将构建软件模块之间的缺陷流。因此，将开发者信息纳入依赖网络对于更全面地理解软件缺陷信息是必要的。

此外，先前的研究通过网络结构分析提取的依赖网络指标需要人工定义，不能全面智能地覆盖所有有用的缺陷信息。由于图神经网络（Graph Neural Network，GNNs）可以通过学习邻接关系和图结构自动编码节点特征为统一维度向量，因此越来越多的研究者利用 GNNs 提高 SDP 模型的准确性[7-8]。正如前文所说，这些利用 GNNs 的方法大多只学习了代码视图的图结构，缺乏利用其他角度的信息学习缺陷相关的特征。

受上述两个方面的启发，本文提出了一种新的通过学习开发者一致性依赖网络的缺陷预测模型（Developer Consistency Dependency Network for Software Defect Prediction，DCN4SDP）。具体来说，首先通过计算代码模块之间的开发者一致性信息构建开发者一致性依赖网络（Developer Consistency Dependency Network，DCN），并且根据代码信息和开发者信息提取了网络结点的初始特征表示；然后使用双向门控图神经网络（Bidirectional Gated Graph Neural Network，BiGGNN）学习图结构上的节点特征，提高了模型自动学习节点特征的能力。DCN4SDP 视图从另一视角解决缺陷预测的问题，将蕴含开发者信息的网络与代码信息的度量元相结合，并智能生成指标。实验结果表明，DCN4SDP 的预测性能相比经典的机器学习分类器和现有的深度学习方法有显著提升。

本文第 2 章介绍了相关工作；第 3 章详细说明了本文提出的方法 DCN4SDP；第 4 章概述了实验设置；第 5 章展示了实验结果；最后总结全文并展望未来。

2　相关工作

2.1　关于构建各种软件依赖图的研究

现有的软件依赖图依赖程序代码视图进行构建。例如，Zimmermann 等[5]基于数据和调用依赖构建了一个依赖网络，以提高 SDP 的性能；Phan 等[3]使用控制流图从代码中提取更深层的语义特征。然而，他们仅专注于从代码视图构建网络，忽略了开发者在软件开发中的影响。此外，手工制作的网络指标受到人为标准的限制，不能全面智能地学习特征。与上述方法不同，本文提出了基于开发者一致性的程序依赖图，并且使用先进的 GNN 模型快速且智能地从图中学习缺陷与模块之间的关系。

2.2　关于 GNN 在 SDP 中的应用研究

由于 GNN 在知识图表示中的强大能力，GNN 及其变体已逐渐成为 SDP 中的热门模型。例如，之前的研究已将图卷积网络（Graph Convolutional Network，GCN）[9]、图注意力网络（Graph Attention Network，GAT）[10]和门控图序列神经网络（Gated Graph Sequence Neural Network，GGNN）[11]应用于软件缺陷预测中以提高性能。然而，这些研究仅将 GNN 应用于由程序代码视图构建的依赖图。此外，传统的 GNN 及其变体将图视为无向图，忽略了重要的方向信息。因此，在其他视图和双向边缘上使用增强的 BiGGNN 来学习节点嵌入，以帮助提取更全面的图信息。

3　本文方法

图 1 给出了 DCN4SDP 的概览，主要包括开发者一致性依赖网络的构建和增强的双向门控图神经网络两部分。

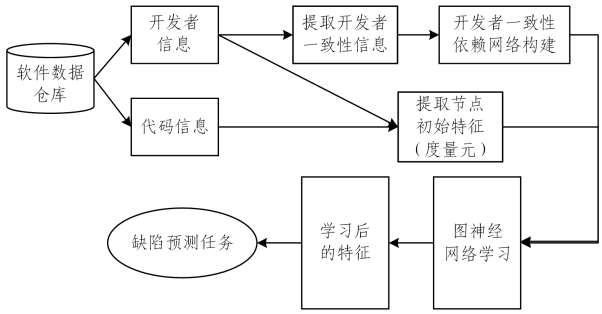

图 1　DCN4SDP 方法的概览

Fig. 1　Overview of DCN4SDP

3.1　开发者一致性依赖网络的构建

建立开发者一致性依赖网络（DCN）的初衷是为了将两个不同的代码模块通过开发者建立起关系，从开发过程中开发人员的角度提取缺陷相关的特征。相较于依赖代码信息的方法，本文方法提供了一个全新的视角来理解模块与模块之间的相互作用。

DCN 的构建是基于开发者信息。开发者信息提取自模块间的开发者关联关系，即如果任意两个模块间存在开发者关联（两个模块是同一个开发者），则表明两个模块节点之间存在联系。也就是说，只要两个模块由同一个开发者创建，则在 DCN 中这两个模块之间就有边关系。

由此，DCN 为一个有向图 G，形式定义为 $G=(N,E)$，其中 N 是模块节点的集合，E 是有向边的集合，如果模块 A 与模块 B 之间存在开发者的直接联系，并且模块 B 的创建时间早于模块 A，则认为模块 A 对 B 存在一个“开发者一致性依赖”，记为 $(A,B)\in E$。模块 A 对 B 的开发者一致性依赖权重 $W_{\mathrm{DCN}}(A,B)$ 的定义如下：

$$W_{\mathrm{DCN}}(A,B)=\begin{cases}\dfrac{|\&d(A),d(B)|}{|d(A)|+|d(B)|}, & t(A)>t(B)\\ 0, & t(A)<t(B)\end{cases} \tag{1}$$

其中，$|d(A)|$ 表示模块的开发者数量，$|\&d(A),d(B)|$ 表示模块 A 与模块 B 相同开发者的数量，$t(A)$ 表示模块的创建时间。只有在模块 B 的创建时间早于模块 A 时式(1)才存在有效值。例如，模块 B 创建时间先于模块 A，A 的开发者数量为

3，B 的开发者数量为 6，且 A 与 B 相同开发者的数量为 3，则最终在 DCN 中，节点 A 对 B 方向的边权重为 0.33。

图结构构建好后，为每个图结点提取初始特征。初始特征提取自模块的代码信息，包括代码度量元、过程度量元和所有权度量元。具体提取方法见 3.2 节。

3.2 双向门控图神经网络

GNN 是一种用于图数据建模和分析的机器学习模型。具体来说，GNN 通过迭代地聚合和更新节点的特征表示，使得模型能够捕捉到图结构中的复杂关系和全局信息，从而在各种任务中展现出强大的建模和分析能力。在 GNN 的基础上，Chen 等[12]提出了一种新颖的双向门控图神经网络(BiG-GNN)，该网络在处理图结构时以交错的方式从传入和传出边缘学习节点嵌入。

在本文建立的有向图中，节点之间的依赖方向可以为缺陷预测提供重要的信息。与忽略消息传递方向并将图视为有向图的大多数 GNN 变体相比，BiGGNN 利用消息传入和传出两个方向交互，并且利用门控循环单元(gated recurrent unit，GRU)作为更新函数过滤不必要的特征信息，拥有提取更多更有用信息的能力。

具体来说，给定一个有向图 $G=(V,E)$，BiGGNN 从图的传入和传出两个方向学习节点嵌入。特别地，每个节点 $v\in V$ 的初始表示为 $\boldsymbol{h}_v^0\in\boldsymbol{R}^d$，其中 d 是维度长度。与 GNN 类似，将消息传递函数应用于固定数量的跳数，即 K。在每个跳数 $k\leqslant K$，对于节点 v，应用求和聚合函数将一组传入(或传出)相邻节点向量作为输入，并且输出后向(或前向)聚合向量。消息传递的计算式如下：

$$\begin{aligned}\boldsymbol{h}_{N_{\Leftarrow}(v)}^k&=SUM(\{\boldsymbol{h}_u^{k-1},\forall\, u\in N_{\Leftarrow}(v)\})\\ \boldsymbol{h}_{N_{\Rightarrow}(v)}^k&=SUM(\{\boldsymbol{h}_u^{k-1},\forall\, u\in N_{\Rightarrow}(v)\})\end{aligned}\tag{2}$$

其中，$N(v)$ 表示节点 v 的邻居节点集合，$\Leftarrow/\Rightarrow$ 是后向和前向。

然后聚合两个方向的节点表示作为跳数 k 的节点表示。

$$\boldsymbol{h}_{N(v)}^k=FUSE(\boldsymbol{h}_{N_{\Leftarrow}(v)}^k,\boldsymbol{h}_{N_{\Rightarrow}(v)}^k)\tag{3}$$

这里的融合函数被设计为两个输入的门控和：

$$\begin{aligned}FUSE(\boldsymbol{a},\boldsymbol{b})&=\boldsymbol{z}\odot\boldsymbol{a}+(1-\boldsymbol{z})\odot\boldsymbol{b}\\ \boldsymbol{z}&=\sigma(\boldsymbol{W}_z[\boldsymbol{a};\boldsymbol{b};\boldsymbol{a}\odot\boldsymbol{b};\boldsymbol{a}-\boldsymbol{b}]+\boldsymbol{b}_z)\end{aligned}\tag{4}$$

其中，$\odot$ 为分量乘法(component-wise multiplication)，σ 是 sigmoid 函数，$\boldsymbol{z}$ 是一个门控向量。最后，将结果提供给 GRU，通过聚合信息更新节点表示。

$$\boldsymbol{h}_v^k=GRU(\boldsymbol{h}_v^{k-1},\boldsymbol{h}_{N(v)}^k)$$

经过 K 跳的计算后，获得最终的节点表示 $\boldsymbol{h}_v^k$。

3.3 分类

首先获取从模型中提取的特征，这些特征通常是高维的向量。为了将这些高维特征有效地映射到最终的类别概率，采用了一个多层感知机(Multi-layer perceptron，MLP)作为分类器。MLP 是一种由多个全连接层构成的神经网络结构，每一层都包含一定数量的神经元，并通过激活函数进行非线性转换，从而能够捕捉数据中复杂的模式和关系。

为了优化这一网络，选择焦点损失(Focal Loss)作为损失函数。焦点损失是交叉熵损失的改进版本，它通过调整每个样本的权重来解决类别不平衡问题。具体来说，它为每个样本引入了一个调整因子，这个因子随着样本被正确分类的概率的增加而增大。因此，对于那些容易分类的样本，它们的损失贡献会减小；而对于难以分类的样本，由于其损失贡献相对较大，模型会被激励去更加关注它们。

焦点损失特别适合处理那些类别极度不平衡的数据集，例如在某些类别的样本非常少的情况下，传统的损失函数可能导致模型偏向于多数类，从而忽视少数类的重要性。本文选择的数据集的缺陷率均小于 20%，这表明样本存在严重的数据不平衡性。通过焦点损失，模型的训练过程更加聚焦于那些难以正确分类的少数类样本，从而提高整体的分类性能。

4 实验设置

4.1 数据集

根据 Yatish 等[13]的建议，选择了 6 个知名的开源软件项目，包括 ActiveMQ，Camel，Derby，Groovy，Hive，JRuby，Lucene 和 Wicket，这些项目代表了一个精心策划的基准缺陷数据集列表，由 16 个版本的数据集组成。这些项目广受欢迎(stars>1.0k)，而且开发时间较长。表 1 列出了本文所研究项目的概览。我们选择的数据集在文件数、缺陷数、代码行数、开发人员数等维度表现出多样性，可以让模型适应不同数据类型，学习到更广泛的特征和模式，从而提高其泛化能力，并且在面对噪声、变化和异常数据时表现更好，提升模型鲁棒性。

表 1 本文所研究数据集的概览

Table 1 Overview of the studied datasets in this paper

项目	描述	版本	文件数	文件缺陷率/%	缺陷数
Active-MQ	开源消息服务器	5.0.0	1 869	15.07	603
		5.1.0	1 943	7.56	287
		5.2.0	2 011	10.59	478
		5.3.0	2 011	10.65	681
		5.8.0	2 339	5.88	346
Camel	企业集成框架，用于集成生产和消费数据的系统	1.4.0	1 481	18.46	430
		2.9.0	7 020	2.91	518
		2.10.0	7 805	2.13	581
		2.11.0	8 714	2.71	464
Groovy	一种编程语言	1.5.7	559	3.44	111
Hive	一个建立在 Hadoop 之上的数据仓库软件	0.9.0	1 311	18.85	377
JRuby	JVM 上的 Ruby 编程语言	1.1	689	11.92	173
		1.4	915	18.75	376
		1.5	1 063	15.07	180
Lucene	文本搜索引擎库	2.3.0	798	24.72	428
		2.9.0	1 363	20.03	540

4.2 网络构建和特征初始化

本节构建 DCN 依赖网络图，网络的结点由代码文件构成，边为两个文件之间开发者一致性依赖的权重。我们通过代码提交和缺陷报告确定文件之间的开发者关联，随后严格按照 2.1 节的内容进行网络的构建。

接下来，获取了对应数据集中的源代码及其对应的提交报告和缺陷报告，并且对所有数据集提取了 54 个代码度量元、5 个过程度量元和 6 个所有权度量元，共 65 个度量元作为网络结点的初始特征。这些研究指标的更详细解释可参考 Yatish 等的研究[13]。将这 3 种度量结合使用，以确保从多个

角度对软件进行全面的特征提取，获得更丰富的信息。这些度量元涵盖了代码信息和开发者信息，再结合我们的开发者依赖网络，使得本文模型可以从代码和开发者两个视角学习特征信息。

根据以往研究的建议[14]对度量元进行标准化。具体来说，按照度量元的类型对同一版本下的度量元值进行标准化，使用这些标准化的指标作为图节点的初始特征进行学习。

据此，在本文的数据集中，图结点为项目文件，边为文件之间开发者的依赖关系，节点初始特征为提取的标准化后的65个度量元。

4.3　数据分割和模型构建

训练集、验证集和测试集都来自项目的同一版本。本文随机独立地从同一版本中选择70%，15%和15%的节点特征作为训练集、验证集和测试集。使用训练集训练模型，使用验证集验证模型，并选择在验证集上表现最好的模型，使用测试集计算模型性能指标。实验过程中，随机重新划分数据集50次，这一过程使我们能够考虑到机器学习模型固有的随机性，并确保结果具有普适性[15]。

本文选择了2个经典机器学习分类器以及1个现有的神经网络模型作为软件缺陷预测基线模型。经典机器学习分类器包括Logistic regression（LR）[16]和Random Forests（RF）[17]，这两个分类器均是成熟使用的机器学习分类器，已被广泛应用于缺陷预测工作中。神经网络模型为DBN[18]，它利用AST提取代码的语义特征，相较于先进的缺陷预测模型取得了较好的效果。本文实验中采用了原文中给定的最优参数作为超参数的选择。

4.4　实验设置

实验中，使用Adam优化器，学习率设置为0.001。将最大epoch的数量设置为200，当没有改善时停止训练过程。神经网络的跳数设置为2，batch size设置为16，少数类新增的采样数为之前数量的两倍。在验证集中使用最优loss度量作为保存最佳模型的基础并进行测试。所有的实验均在2块80GB的NVIDIA Tesla A800上运行。

4.5　模型评估

为了评估分类模型的有效性，我们使用了ROC曲线下与坐标轴围成的面积（Area Under the Receiver Operating Characteristic Curve，AUC）[19]、召回率（Recall）、布里尔分数（Brier score）[20]、错误报警概率（the Probability of False Alarm，PF）和F1分数（F1-score）作为评价指标。通过这些指标，能全面地评估分类模型的表现，确保模型在不同的应用场景下达到预期的效果。

此外，使用威尔科克森符号秩检验（Wilcoxon-signed rank test）[21]和克利夫Delta效应量检验（Cliff's Delta effect size test）[22]计算两个不同模型之间每个测量上统计的显著差异。Wilcoxon-signed rank test通常用于成对比较，这是一种非参数测试方法，用于确定两组匹配的样本是否具有相同的分布。p值可以用来确定两组匹配的样本之间的统计学差异是否显著，如果p值<0.05，则在95%置信水平下存在统计显着性差异（L）。此外，由于执行了多个成对比较，因此使用Bonferroni校正[23]从Wilcoxon符号秩检验中获得的p值以控制误报。Cliff's Delta侧重于比较两组模型之间的测量差异，如果Cliff's Delta $\geqslant 0.33$，则两组模型之间在统计上存在较大差异（L）。

5　实验结果

本章研究了DCN4SDP的性能，不仅将其与其他现有缺陷预测模型的性能进行了比较，还评估了其相较于传统机器学习分类器模型性能方面的改进效果。

表2列出了DCN4SDP与基线模型在AUC，Recall，Brier，PF和F1方面的比较结果。对6个项目的16个版本数据分别进行了50次实验，并取平均值，然后将每个项目的结果再次取平均值记录在表格中，最后统计了所有项目的平均结果。其中每个度量的最佳结果都以粗体标记。表3列出了DCN4SDP和基线之间16个版本的每个测量值的平均值效应大小。将Wilcoxon-signed rank test和Cliff's Delta effect size test的结果存在统计上的较大差异的值以粗体标记。

表2　各方法的平均实验结果

Table 2　Average experimental results of various methods

项目	方法	AUC	Recall	Brier	PF	F1
ActiveMQ	DCN4-SDP	**0.91**	**0.70**	**0.05**	**0.02**	**0.74**
	DBN	0.66	0.62	0.23	0.15	0.67
	LR	0.68	0.21	0.08	0.02	0.30
	RF	0.88	0.34	0.06	0.03	0.45
Camel	DCN4-SDP	**0.98**	**0.88**	**0.01**	**0.01**	**0.90**
	DBN	0.65	0.72	0.22	0.16	0.74
	LR	0.53	0.11	0.06	0.01	0.16
	RF	0.86	0.22	0.04	0.01	0.33
Groovy	DCN4-SDP	**0.87**	**0.68**	**0.02**	0.02	0.50
	DBN	0.70	0.49	0.23	0.14	**0.72**
	LR	0.58	0.30	0.04	0.01	0.34
	RF	0.79	0.24	0.03	0.01	0.38
Hive	DCN4-SDP	0.91	**0.70**	**0.08**	**0.04**	**0.86**
	DBN	0.65	0.68	0.22	0.20	0.72
	LR	0.82	0.33	0.11	0.03	0.45
	RF	**0.91**	0.59	0.08	0.06	0.67
Jruby	DCN4-SDP	**0.94**	**0.78**	**0.05**	0.04	**0.84**
	DBN	0.66	0.71	0.21	0.15	0.74
	LR	0.67	0.36	0.13	0.04	0.45
	RF	0.89	0.51	0.08	**0.03**	0.61
Lucene	DCN4-SDP	0.87	**0.73**	**0.09**	**0.04**	**0.73**
	DBN	0.69	0.67	0.20	0.14	0.66
	LR	0.68	0.25	0.17	0.07	0.33
	RF	**0.89**	0.55	0.11	0.10	0.61
平均值	DCN4-SDP	**0.91**	**0.74**	**0.05**	**0.03**	**0.76**
	DBN	0.67	0.65	0.22	0.16	0.71
	LR	0.66	0.26	0.10	0.03	0.34
	RF	0.87	0.41	0.07	0.04	0.51

从表2可以看出，DCN4SDP在所有指标上均取得了比其他模型更好的成绩。具体而言，AUC值达到0.91，相比最接近的RF模型的0.87有明显提升，远超过DBN的0.67和LR的0.66，表明DCN4SDP在区分正负样本的能力上具有显著优势。在召回率（Recall）方面，DCN4SDP同样表现优异，达到0.74，是所有模型中最高的。它比DBN的0.65略高，而与RF的0.41和LR的0.26相比则显得尤为突出，表明DCN4SDP在正确识别正例方面更有效。Brier分数作为预测准确性的一个指标，DCN4SDP的得分为0.05，低于DBN的0.22、LR的0.10和RF的0.07，这一结果说明了

DCN4SDP 在预测精度上的显著优势。误报率(PF)越低,说明模型在避免错误标记非缺陷模块方面表现越好。DCN4SDP 的 PF 值为 0.03,等于或低于所有其他模型,包括 LR 的 0.03、DBN 的 0.16 及 RF 的 0.04,进一步证明了其在减少误报方面的性能。最后,DCN4SDP 的 F1 为 0.76,同样超过所有其他方法,包括 DBN 的 0.71、RF 的 0.51 和 LR 的 0.34,这反映了 DCN4SDP 在保持精确度和召回率之间良好平衡的能力。

进一步地,通过统计测试增强了对模型性能的理解。从表 3 中观察到了 DCN4SDP 与其他模型之间的显著性差异。除了与 RF 和 LR 在 PF 指标上没有显著差异外,Cliff 的 Delta 检验的结果全为"large",Wilcoxon 符号秩检验结果也支持这一发现,与其他所有分类器的所有指标相比都显示了"large"的效应量。并且 DCN4SDP 的 PF 值也展现出了极低的水平,并不逊色于 RF 和 LR。这些统计显著性检验的结果不仅证实了 DCN4SDP 在各个评价指标上的卓越表现,也反映了其在实际应用中相对于其他方法的优越性。

表 3 DCN4SDP 与其他方法的统计学分析对比

Table 3 Statistical analysis comparison between DCN4SDP and other methods

		AUC	Recall	Brier	PF	F1
DBN	p-value	9.02×10^{-8}(L)	7.92×10^{-3}(L)	8.56×10^{-10}(L)	1.48×10^{-10}(L)	1.84×10^{-4}(L)
	delta	0.894(L)	0.876(L)	0.910(L)	0.556(L)	0.548(L)
LR	p-value	1.46×10^{-13}(L)	9.52×10^{-3}(L)	8.66×10^{-7}(L)	2.17×10^{-1}	3.55×10^{-7}(L)
	delta	0.937(L)	0.922(L)	0.647(L)	0.182	0.910(L)
RF	p-value	3.17×10^{-2}(L)	1.84×10^{-2}(L)	6.03×10^{-2}(M)	1.13×10^{-1}	6.06×10^{-8}(L)
	delta	0.677(L)	0.666(L)	0.490(L)	0.249	0.774(L)

结束语 依赖网络度量元在软件缺陷预测(SDP)中的有效性已得到广泛的研究和验证。然而,目前大多数依赖网络度量元主要依赖于程序代码信息来构建网络。由于软件是复杂的人工产品,除代码外的许多人为因素也极大地影响着软件缺陷的形成,如开发者的编程习惯和协作模式,仅依赖于代码信息的提取显然无法全面地捕获影响软件质量的全部因素。

此外,传统的度量元通常是手工设计的,这些度量往往受限于人为设定的标准和模型,未能充分利用机器学习技术自动生成特征的潜力。依赖于手工提取的度量元通常难以全面且深入地理解和解决问题。

针对上述问题,本文提出了一种新的缺陷预测模型——DCN4SDP,该模型利用图神经网络从开发者一致性依赖网络中学习有效的缺陷预测信息。DCN4SDP 首先整合了开发者之间的关联关系,并结合代码层面的度量元,通过这种多维度的视角来构建更为全面的依赖网络。然后,模型采用了先进的双向门控图序列神经网络(BiGGNN)技术,实现对软件特征的自动化学习,从而提高预测的准确性和效率。

在多个开源软件项目的 16 个不同版本的数据集上评估了 DCN4SDP 模型的性能,并将其与现有的主流缺陷预测方法以及两个常用的机器学习分类器进行了比较。实验结果显示,DCN4SDP 在多个关键性能指标上均表现出色,尤其是在 AUC,Recall 和 F1 得分方面展现出了优于传统方法的性能。这一结果不仅证明了开发者依赖网络分析在软件缺陷预测中的有效性,也展示了图神经网络在处理此类问题上的强大潜力。

然而,本文方法还存在一些可以改进的地方,例如,新型的软件缺陷预测模型 DCN4SDP 在多个标准数据集上表现良好,但其泛化能力和适用范围仍需要进一步验证;开发者个人特质可能是主观和难以量化的,因此在构建开发者一致性依赖网络时,可能存在主观性和主观偏见,影响模型的准确性和稳定性。在未来的研究中,将致力于探索 DCN4SDP 在更多缺陷预测领域(如跨版本、跨项目、跨公司缺陷预测)上的表现,并且进一步优化模型,以减小主观偏见的影响,提升其在不同开发环境中的稳定性和泛化能力。

参 考 文 献

[1] ZAIN Z M, SAKRI S, ISMAIL N H A. Application of deep learning in software defect prediction: systematic literature review and meta-analysis [J]. Information and Software Technology, 2023, 158: 107175.

[2] TIAN X, CHANG J, ZHANG C, et al. Survey of open-source software defect prediction method[J]. Journal of Computer Research and Development, 2023, 60(7): 1467-1488.

[3] QIU S, HUANG M, LIANG Y, et al. Code multiview hypergraph representation learning for software defect prediction[J]. IEEE Transactions on Reliability, 2024, 73(4): 1863-1876.

[4] PHAN A V, NGUYEN M L, BUI L T. Convolutional neural networks over control flow graphs for software defect prediction [C]// Proceedings of the 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). Boston, USA: IEEE, 2017: 1-8.

[5] ZIMMERMANN T, NAGAPPAN N. Predicting defects using network analysis on dependency graphs[C]// Proceedings of the 30th international conference on Software engineering. Leipzig, Germany, 2008: 531-540.

[6] OSTRAND T J, WEYUKER E J, BELL R M. Programmer-based fault prediction[C]// Proceedings of the 6th International Conference on Predictive Models in Software Engineering. Timisoara, Romania, 2010: 1-10.

[7] TANG F, HE P. Software Defect Prediction using Multi-scale Structural Information [C]// Proceedings of the 2023 9th International Conference on Computing and Artificial Intelligence. 2023: 548-556.

[8] MA J, SUN Y Y, HE P, et al. GSAGE2defect: An Improved Approach to Software Defect Prediction based on Inductive Graph Neural Network [C]// Proceedings of the International Confer-

ence on Software Engineering and Knowledge Engineering (SEKE). 2023：45-50.

[9] ZENG C，ZHOU C Y，LV S K，et al. Gcn2defect：Graph convolutional networks for smotetomek-based software defect prediction[C]// Proceedings of the 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE). Wuhan，China：IEEE，2021：69-79.

[10] XU J，WANG F，AI J. Defect prediction with semantics and context features of codes based on graph representation learning [J]. IEEE Transactions on Reliability，2020，70(2)：613-625.

[11] ZHOU C，HE P，ZENG C，et al. Software defect prediction with semantic and structural information of codes based on graph neural networks [J]. Information and Software Technology，2022，152：107057.

[12] CHEN Y，WU L，ZAKI M J. Reinforcement learning based graph-to-sequence model for natural question generation [EB/OL]. (2019-08-14) [2024-04-19]. http://arxiv.org/abs/1908.04942.

[13] YATISH S，JIARPAKDEE J，THONGTANUNAM P，et al. Mining software defects：Should we consider affected releases? [C]// Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). Montreal，Canada：IEEE，2019：654-665.

[14] WANG S，LIU T，NAM J，et al. Deep semantic feature learning for software defect prediction[J]. IEEE Transactions on Software Engineering，2018，46(12)：1267-1293.

[15] XUAN J，JIANG H，HU Y，et al. Towards effective bug triage with software data reduction techniques[J]. IEEE Transactions on Knowledge and Data Engineering，2014，27(1)：264-280.

[16] MALHOTRA R. A systematic review of machine learning techniques for software fault prediction[J]. Applied Soft Computing，2015，27：504-518.

[17] BREIMAN L. Random Forests[J]. Machine Learning，2001，45：5-32.

[18] WANG S，LIU T，TAN L. Automatically learning semantic features for defect prediction[C]// Proceedings of the 38th International Conference on Software Engineering. Austin，USA，2016：297-308.

[19] LESSMANN S，BAESENS B，MUES C，et al. Benchmarking classification models for software defect prediction：A proposed framework and novel findings[J]. IEEE Transactions on Software Engineering，2008，34(4)：485-496.

[20] RUFIBACH K. Use of Brier score to assess binary predictions [J]. Journal of Clinical Epidemiology，2010，63(8)：938-939.

[21] WOOLSON R F. Wilcoxon signed-rank test[J/OL]. https://doi.org/10.1002/0470011815.b2a15177.

[22] MACBETH G，RAZUMIEJCZYK E，LEDESMA R D. Cliff's delta calculator：a nonparametric effect size program for two groups of observations [J]. Universitas Psychologica，2011，10(2)：545-555.

[23] ARMSTRONG R A. When to use the Bonferroni correction [J]. Ophthalmic and Physiological Optics，2014，34(5)：502-508.

QIAO Yu，born in 1999，postgraduate，is a member of CCF (No. R4417G). His main research interests include intelligent software engineering and software defect prediction.

XU Tao，born in 1978，postgraduate，senior experimenter. His main research interests include neural networks，distributed storage，and software defect prediction.

（责任编辑：何杨）