|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 论文题目 | DeepLineDP: Towards a Deep Learning Approach for Line-Level Defect Prediction | | | |
| Paper URL | <https://ieeexplore.ieee.org/abstract/document/9689967> | | | |
| Project URL | <https://github.com/awsm-research/DeepLineDP> | | | |
| 综述/背景介绍 | 发展状况 | 原因 | 意义 | 关键词（速记词汇、信息索引词汇） |
| 最近的研究发现，自动学习句法和语义特征的深度学习方法优于使用过程和产品作为特征度量的传统机器学习方法。但是之前的研究中存在两个主要的局限性：（1）基于深度学习的缺陷预测的颗粒度水平仍然是粗粒度的。比如类和方法级别（2）周围的token和周围行还没有得到充分的利用。 | | | 深度学习；语义 |
| 假设 |  | | |  |
| 方法描述(含图) | DeepLineDP通过使用双向GRU单元来捕获有缺陷的代码行，使用分层注意网络来学习周围的标记和周围的行。从数据收集和数据准备开始，以生成包含源代码文件、文件级和行级基础真实标签（即受发布后缺陷影响的文件和为解决缺陷而改变或删除的行）的数据集。对于每个源代码文件，我们执行先执行代码预处理，包括代码抽象化和词汇量管理。然后进行源代码表示，生成一个向量表示。之后使用分层注意力网络来学习源代码的分层结构，以进行文件级的缺陷预测，同时实现行级的缺陷预测。然后，预测层产生预测缺陷文件的概率分数。最后使用token级注意力层，根据对缺陷文件预测有贡献的最重要的token来识别缺陷行。（图3） | | | 双向GRU，分层注意网络 |
| 实验设计 | 代码行缺陷数据集：由32个软件版本组成，横跨9个开源软件系统。  训练详情:单词嵌入向量大小d：50;学习率：0.001  batch size：32;  超参数调整:  Bi-GRU hidden cells of token encoder={32, 64, 128}.  Bi-GRU hidden cells of line encoder = {32, 64, 128}.  MLP hidden cells of token attention = {64, 128}.  MLP hidden cells of line attention = {64, 128}.  learning rate ={0.01, 0.001, 0.0001}.  评价指标：  文件级别：AUC、平衡精度和MCC  代码行级别：Recall@Top20%LOC，Effort@Top20%Recall，Initial False Alarms。 | | |  |
| 数据处理 | 输入 | 筛除特例 | 处理方式 | 关键词（速记词汇、信息索引词汇） |
| Java文件 |  | Java文件解析成多行，每行解析成多个token | 代码token |
| 结论 | DeepLineDP是一种深度学习方法，用于自动学习周围标记和行的语义属性，以识别有缺陷的文件和有缺陷的行。通过对9个软件项目的32个版本的案例研究，发现代码token的风险得分因其位置不同而有很大差异。我们的DeepLineDP比其他文件级缺陷预测方法的准确率高14%-24%；比其他行级缺陷预测方法的成本效益高50%-250%；在转移到其他软件项目时，也能达到合理的性能。 | | | |
| 局限性分析 | **结构有效性威胁：**与评估方法得适用性有关。**外部有效性威胁：**研究结果仅限于9个软件项目的32个软件版本，不具有普遍性。**内部有效性威胁：**超参数设置 | | | |

2.论文总结：

**（论文名）：**DeepLineDP: 用于行级别的缺陷预测的深度学习方法

**（题目）**：DeepLineDP: Towards a Deep Learning Approach for Line-Level Defect Prediction

**（论文URL）：**<https://ieeexplore.ieee.org/abstract/document/9689967>

**（项目URL）：**<https://github.com/awsm-research/DeepLineDP>

**（总结）：**本文提出了DeepLineDP，它是一种深度学习方法，用于学习周围的代码标记和源代码的周围行，以预测有缺陷的文件和定位有缺陷的行。目前缺陷预测方法没有很好的利用周围的token和周围行，并且还是粗粒度的缺陷预测，开发人员仍需花费大量时间去定位缺陷行，因此提出了DeepLineDP，该方法首先将每个代码token生成一个向量表示，然后采用双向GRU单元捕获源代码行的周围token，再采用层次注意网络（HAN）学习源代码的层次机构以越策有缺陷的文件，之后利用注意力机制计算出有助于预测特定缺陷文件的代码token的风险分数，再使用出现在该行中的代码token的风险得分之和，生成一个风险行的排名。通过对9个软件项目的32个版本的案例研究,发现代码token的风险得分因其位置不同而有很大差异。我们的DeepLineDP比其他文件级缺陷预测方法的准确率高14%-24%；比其他行级缺陷预测方法的成本效益高50%-250%；在转移到其他软件项目时，也能达到合理的性能。

**（附图）：**

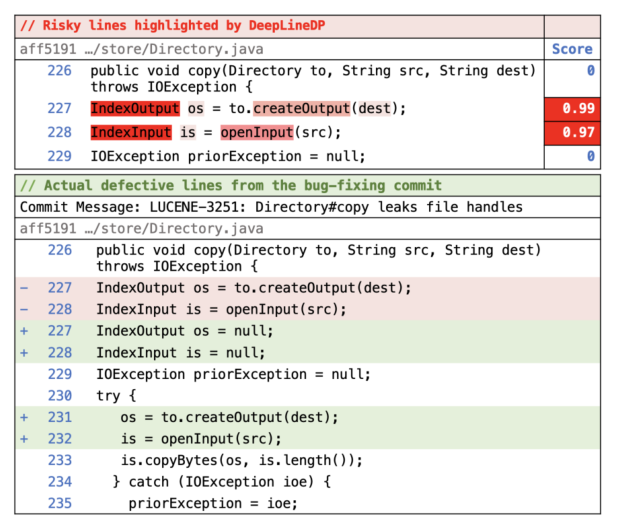


图1：一个可视化的例子，突出了从我们的DeepLineDP方法获得的最有风险的token。

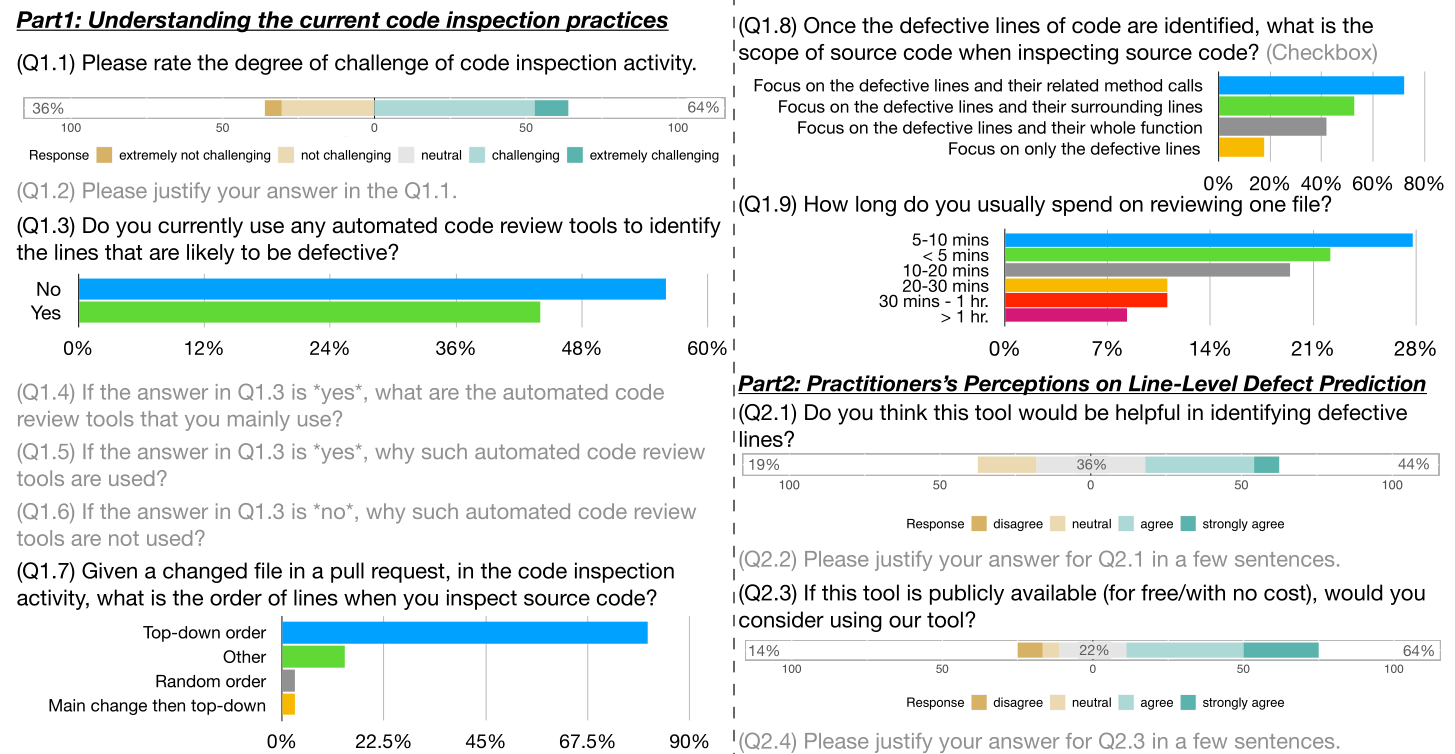


图2：（MQ1/MQ2）调查问题的摘要和从36名参与者那里得到的结果。

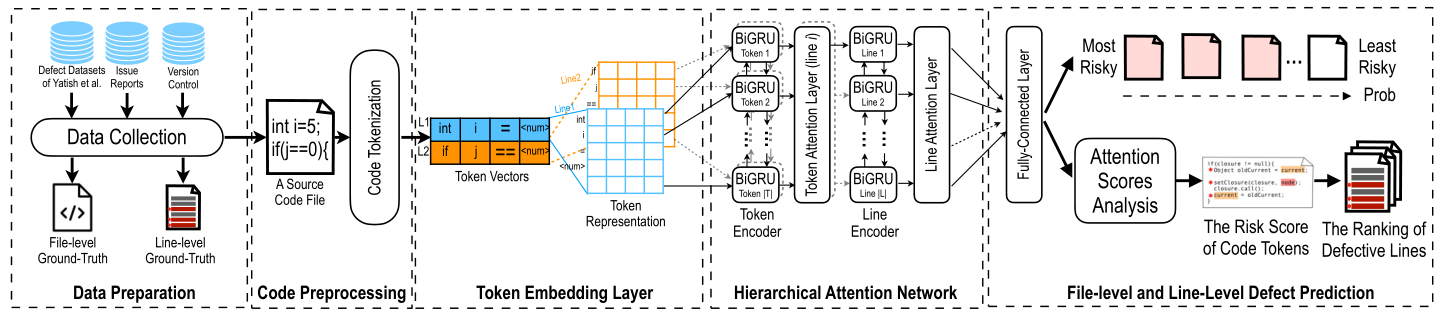


图3：我们的DeepLineDP方法的概览图。

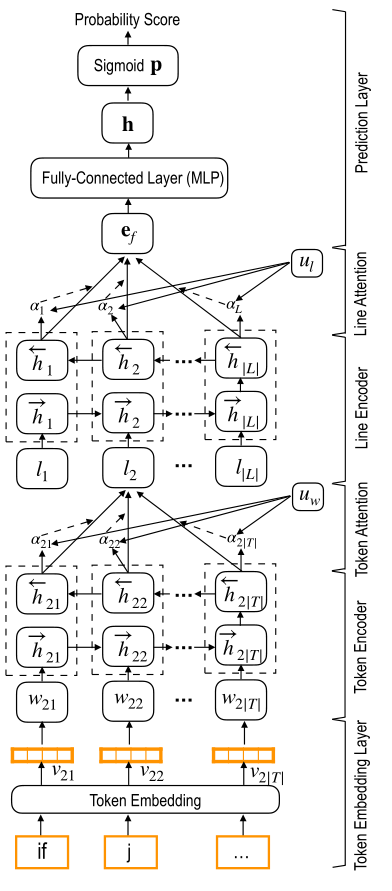


图4：一个两层的分层注意网络的概述

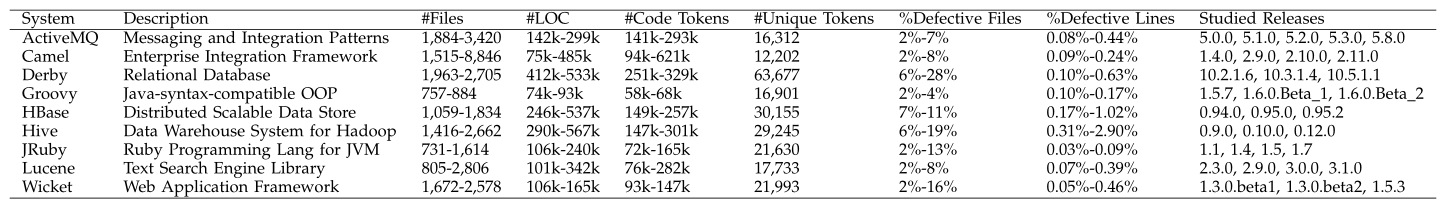


表1: 研究项目的概述。

图5：（RQ1）缺陷行和干净行中的代码标记的风险分数的变化。

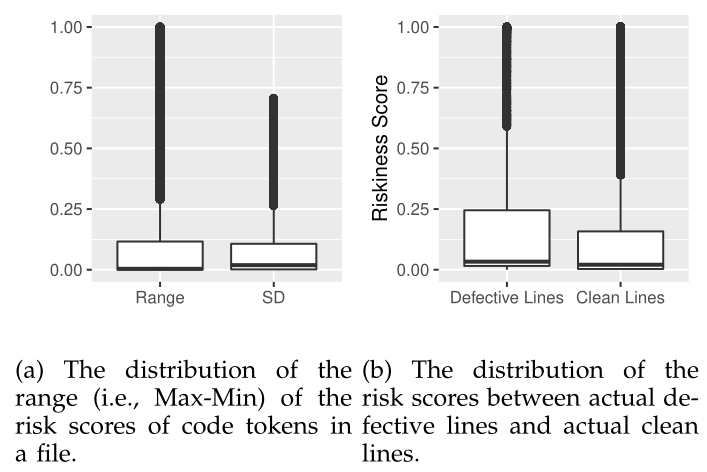


图5：（RQ1）缺陷行和干净行中的代码Token的风险分数的变化。

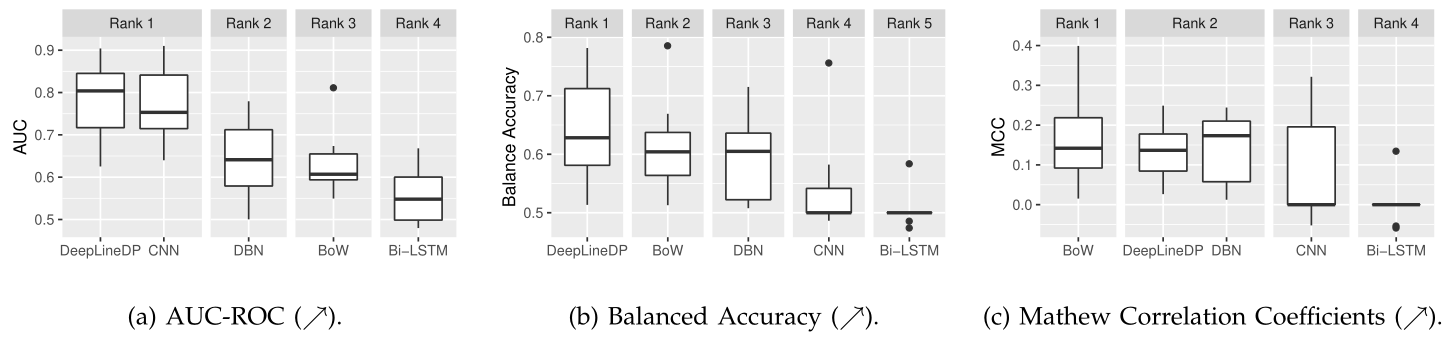


图6：（RQ2）ScottKnott ESD排名以及我们的DeepLineDP和最先进的文件级缺陷预测方法的AUC、平衡准确率和Mathew相关系数（MCC）的分布。

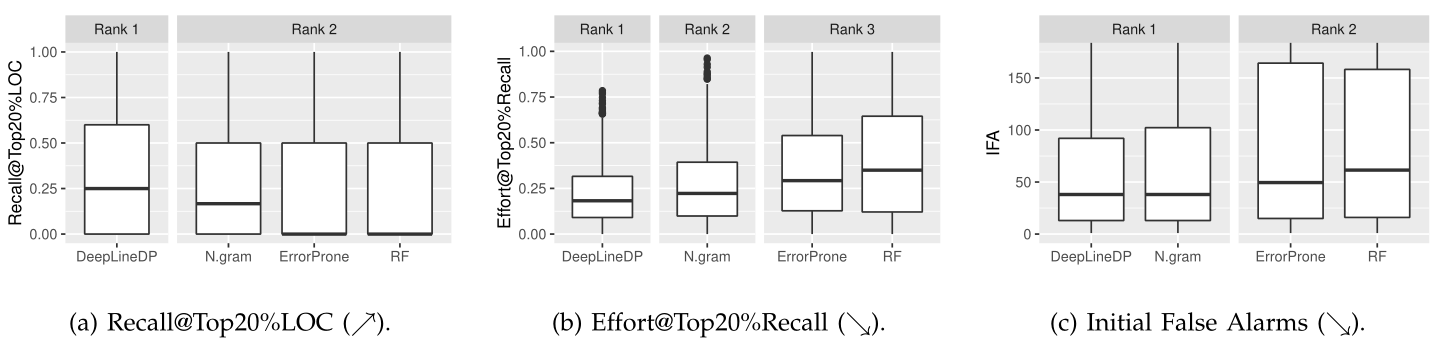


图7：（RQ3）ScottKnott ESD排名以及我们的DeepLineDP和最先进的行级别缺陷预测方法的Recall@Top20%LOC, Effort@Top20%Recall, and Initial False Alarms 的分布。

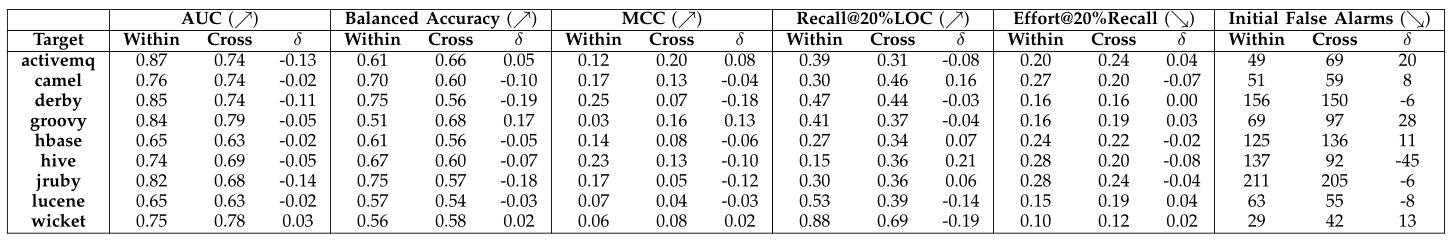


表2：（RQ4）我们的DeepLineDP方法在项目内（即使用用自己的项目训练的模型）和跨项目评价（即使用用其他项目训练的模型）的平均性能及其百分点差异