软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(5):1924−1948 [doi: 10.13328/j.cnki.jos.007178] [CSTR: 32375.14.jos.007178]

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

结合主动学习和半监督学习的软件可追踪性恢复框架*

董黎明[1,2], 张 贺[1,2], 孟庆龙[1,2], 匡宏宇[1,2]

[1](南京大学 软件学院, 江苏 南京 210093)

[2](计算机软件新技术国家重点实验室 (南京大学), 江苏 南京 210093)

通信作者: 张贺, E-mail: hezhang@nju.edu.cn

摘 要: 软件可追踪性被认为是软件开发过程可信的一个重要因素, 确保对软件开发过程的可见性并进行全面追踪, 从而提高软件的可信度和可靠性. 近年来, 自动化的软件可追踪性恢复方法取得了显著进展, 但在企业项目中的应用仍面临挑战. 通过调研研究和实验案例分析, 发现工业界场景中可追踪性模型表现不佳的 3 个关键挑战: 原始数据低质量、样本稀疏性和不平衡性, 并提出一种结合主动学习和半监督学习的软件可追踪性恢复框架 STRACE(AL+SSL). 该框架通过选择有价值的标注样本和生成高质量的伪标签样本, 有效利用未标注的样本, 克服数据低质量和稀疏性挑战. 实验基于 10 个样本规模在几万至近百万个 issue-commit 跟踪对实例的企业项目, 进行多组对比实验, 结果表明该框架在当前真实企业项目软件可追踪性恢复任务上具有有效性. 其中消融实验结果表明 STRACE(AL+SSL) 中主动学习模块所选择的无标签样本在可追踪性恢复任务中发挥了更为重要的作用. 此外, 还验证各个模块最佳的样本选择策略组合, 包括调整后的半监督类平衡自训练样本选择策略 CBST-Adjust 和低成本高效率的主动学习子模块互信息 SMI_Flqmi 样本选择策略.

关键词: 软件可追踪性; 主动学习; 半监督学习

中图法分类号: TP311

中文引用格式: 董黎明, 张贺, 孟庆龙, 匡宏宇. 结合主动学习和半监督学习的软件可追踪性恢复框架. 软件学报, 2025, 36(5): 1924–1948. http://www.jos.org.cn/1000-9825/7178.htm

英文引用格式: Dong LM, Zhang H, Meng QL, Kuang HY. Software Traceability Recovery Framework Based on Active Learning and Semi-supervised Learning. Ruan Jian Xue Bao/Journal of Software, 2025, 36(5): 1924–1948 (in Chinese). http://www.jos.org.cn/1000-9825/7178.htm

Software Traceability Recovery Framework Based on Active Learning and Semi-supervised Learning

DONG Li-Ming[1,2], ZHANG He[1,2], MENG Qing-Long[1,2], KUANG Hong-Yu[1,2]

[1](Software Institute, Nanjing University, Nanjing 210093, China)

[2](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

Abstract: Software traceability is considered critical to trustworthy software engineering, ensuring software reliability through the tracking of the software development process. Despite significant progress in automatic software traceability recovery techniques in recent years, their application in real-world commercial software projects does not meet expectations. An investigation into the application of learning-based software traceability recovery classifier models in commercial software projects is conducted. It uncovers three critical challenges faced in industrial settings. These challenges contribute to underperforming traceability models: low-quality raw data, data sparsity, and class imbalance. In response to these challenges, STRACE(AL+SSL) is proposed. It is a software traceability recovery framework that integrates active learning and semi-supervised learning. By strategically selecting valuable annotated samples and generating high-quality

* 基金项目: 国家自然科学基金 (62072227, 62202219); 国家重点研发计划 (2019YFE0105500); 江苏省重点研发计划 (BE2021002-2); 南京大学计算机软件新技术国家重点实验室创新项目 (ZZKT2022A25); 海外开放课题 (KFKT2022A09)
收稿时间: 2023-06-01; 修改时间: 2023-08-13, 2023-11-25; 采用时间: 2024-03-14; jos 在线出版时间: 2024-09-04
CNKI 网络首发时间: 2024-09-05

pseudo-labeled samples, STRACE(AL+SSL) effectively harnesses unlabeled data to address data-related challenges. Multiple comparative experiments are conducted with nearly one million issue-commit trace pair samples from 10 different enterprise projects. The results of these experiments validate the effectiveness of the proposed framework for real-world software traceability recovery tasks. The ablation results show that the unlabeled samples selected by the active learning in STRACE(AL+SSL) play a crucial role in the traceability recovery task. Additionally, the optimal combination of sample selection strategies in STRACE(AL+SSL) is confirmed. This includes CBST-Adjust for the semi-supervised sample rebalancing strategy and SMI_Flqmi, which is recognized for its cost-effectiveness and efficiency in active learning.

Key words: software traceability; active learning; semi-supervised learning

软件可追踪性对于管理软件开发过程、降低软件复杂性成本都至关重要[1,2]. 从相关研究中不难发现, 软件可追踪性与不同环节发挥着不同的作用, 例如在微服务架构中维护软件制品间可追踪性有利于拆分服务和识别不再需要的服务[3]. 此外软件可追踪性若缺失则使得开发人员难以明确代码所有权, 从而导致相同的功能被实现多次[4], 软件复杂性和维护成本进一步增加. 在测试环节, 通过分析测试用例中哪些需要重写或简化, 以及增加测试覆盖度, 有助于推荐质量膨胀的需求[5]. 然而, 随着软件项目的演进, 保持高质量的软件可追踪性链接是一个代价高昂的过程[6]. 因此, 相关研究开始关注软件可追踪性的自动化方法[7].

近年来, 软件可追踪性恢复模型大多都基于机器学习方法, 相关研究通过在开源项目或可追踪性领域内公开数据集上的实验取得了较为理想的效果. Mills 等人[8]提出了可追踪性分类模型 TRAIL, 通过组合对比 6 种特征选择算法、4 种数据重采样方法和 6 种分类算法, 总结出了预测制品间跟踪链接的分类模型的最佳配置组合, 即: 使用 Pearson 相关性方法做特征选择, 使用过采样方法 (synthetic minority oversampling technique, SMOTE)[9]做数据重采样, 使用随机森林 (random forest, RF) 作为分类算法可以得到最佳模型效果, 相关结果在 11 个可追踪性领域常用数据集上得到了有效验证. Rath 等人[10]提出了缺陷-commit 跟踪链接恢复和推荐的随机森林模型, 通过挖掘制品之间的人员关系、时间关系和前后追踪关联关系等过程信息, 选择了 16 项过程特征和 2 项文本相似性特征. 通过在 6 个开源项目上展开一系列实验发现, 推荐跟踪链接任务上平均召回率, 即 Top 3 的召回率 *Recall* 达到 96%, 精确率 *Precision* 达到 33%. 恢复跟踪链接任务上 6 个项目的精确率均高于 89%, 召回率平均为 50%. 当然近几年也有一部分研究工作, 逐渐开始应用工业界项目进行相关方法的验证, 也取得了一定进展[11−14]. Guo 等人[13]利用 RNN 模型提出了一个基于深度学习的可追踪性模型, 模型使用了从交通控制系统中收集到的小型数据集进行验证, 用于恢复子系统需求和设计定义之间的链接关系. 此外, 通过多维度度量制品文本相似性, 结合开发人员的反馈和传播链接机制, Moran 等人[12]实现了可追踪性贝叶斯层次化概率模型 (Comet), 他们在 Cisco 公司的一个小型项目上验证了该模型, 并开发了基于 Jenkins 的可追踪性链接推荐插件.

然而, 本文发现, 基于学习式的可追踪恢复技术在企业级项目中的实验效果和应用突破仍然面临挑战. 由于复杂的企业项目中, 软件制品间可追踪性质量问题更为严峻, 因此企业项目对于可追踪性恢复自动化实践需求更为紧迫. 企业项目中, 软件方法、使用语言、组织团队的更新迭代十分频繁, 由此造成软件制品频繁演化, 不同制品 (如需求文档、缺陷报告、测试用例和源代码等) 之间的潜在的跟踪链接规模很大, 高质量的跟踪链接很难通过人为手段恢复和维护. 又因为历史数据中制品间存在的关联的链接比例太低, 导致有标签样本非常稀缺, 所以难以构建有效的制品间的跟踪链接预测模型以提高软件可追踪性质量.

其次, 软件可追踪性是将不同制品进行匹配, 从而判断他们之间是否存在链接, 而“真”“假”链接的比例过于悬殊就会使得样本数据中的不平衡性增加. 例如一个需求或缺陷 (源制品) 可以追溯到一个或多个代码提交记录 (目标制品), 因此源制品生命周期中生成的所有潜在代码提交都被选择为目标制品. 在工业环境中, 往往由于紧张的交付周期使得代码提交更频繁, 则很大比例的与源制品不存在关联关系的代码提交将被选择作为其候选目标制品. 在以往的研究中, 很少对软件可追踪性的数据不平衡问题提出过针对性的解决方案.

主动学习和半监督学习的共性是都利用到了未标注的数据和已标注的数据, 从而提高模型的学习能力. 而真实企业项目中, 可追踪性比例低的问题恰好可以提供较多的未标注数据. 本文在前期相关研究基础上[10], 在可追踪性框架中同时结合了主动学习和半监督学习这两种学习策略, 选择未标注数据更新训练集, 并调整其中的样本

选择策略, 重新平衡训练集分布, 针对性地应对企业项目面临的诸多挑战, 致力于改进工业界场景中可追踪性模型表现不佳的现状.

本文第 1 节介绍软件可追踪性相关方法和研究现状. 第 2 节介绍本文前期案例工作和在企业项目中应用可追溯性模型后分析出来的几点关键挑战等. 第 3 节介绍本文提出的基于结合主动学习及半监督学习改进策略的自动化可追踪性分类模型构建流程. 第 4 节在 6 个开源项目及 10 个企业项目上展开多组对比实验, 验证了所提模型的有效性. 第 5 节讨论本文提出方法面临一些约束和挑战, 并从内部效度、结论效度和外部效度等方面分析本文实验和结论的有效性威胁. 第 6 节在最后总结全文.

1 软件可追踪性相关工作

软件的可追踪性是指“将任何唯一可识别的软件工程制品与其他制品相互关联的能力”[1]. 软件可追踪性被认为是软件开发过程可信一个重要因素. 软件可追踪性可以促进软件系统生命周期内各个阶段的实践任务, 从而提高生产力和质量[15–17]. 在满足软件可追踪性质量前提下, 研究人员和项目实践人员可以利用已知的软件制品间的可追踪性信息, 辅助其他多种研究活动和决策, 如需求重用和验证[18]、变更管理[19]、项目和风险管理[20]、缺陷预测和定位[21]、测试用例优先级排序[11,22]等研究中. 此外, 软件过程制品的可追踪性有助于构建类似于真实世界中软件研发过程的仿真模型, 在开发生命周期早期阶段, 帮助项目经理进行项目规划和识别潜在风险[23,24]. 软件可追踪性是项目管理和质量保障实践中的基本组成要素, 支持开发人员提高软件系统的可维护性和可靠性[25,26].

软件制品是研发过程中的重要的产物, 相互之间通过不同软件活动形成关联关系. 如图 1 所示, 设计人员分析用户需求后, 为开发人员制定每个版本的新需求. 开发人员对分配到的需求进一步拆解、设计、讨论后进行编码. 然后通过 commit 合入代码仓. commit 记录自动地将一个或多个源代码文件捆绑在一起. 因此 commit 和源代码之间的关联关系自然生成. 之后代码会被打包给测试人员执行测试, 测试用例检测到的缺陷结果会被提为缺陷单. 开发人员对各自所负责的需求产生的缺陷单进行修复. 此过程中, 举例来讲, 开发人员需要明确 commit 的标记, 即对应的需求 ID 或缺陷 ID[27], 在 commit 与相关制品之间显式地创建跟踪链接. 如果从缺陷到 commit 至少有一个跟踪链接, 则缺陷被认为是 linked issue. 没有任何链接的缺陷被称为 unlinked issue. 其他制品同理. 然而, 在对软件过程制品进行信息化管理时, 各个阶段软件活动会通过不同的工具和系统进行管理维护. 比如, 在企业内普遍采用在线软件系统, 如需求看板系统 (Kanban), 代码托管平台系统 (如 GitHub) 和缺陷跟踪系统 (如 Jira) 来管理各个软件制品. 因此生成的数据会存储在不同的软件存储库中, 且制品间的关联关系主要依赖人工方式创建, 不仅费力耗时, 还经常存在关联关系缺失的问题, 形成了“数据孤岛”. 研究人员还发现, 由于不同制品 (如需求、缺陷和代码) 之间的潜在跟踪链接规模很大, 以及长时间的迭代开发过程造成软件制品的频繁演化, 高质量的跟踪链接很难通过人工恢复和维护[28].

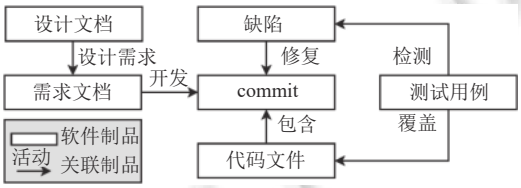


图 1 软件过程制品及其关联关系

近年来, 越来越多的研究开始提出 (半) 自动化的算法模型来帮助开发者自动恢复可追踪性[29–31]. Aung 等人[32]的研究综述中汇总了大约近 10 年的软件可追踪性方法的演进过程. 其中包括基于启发式 (heuristic-based) 的方法、基于概率的方法[33]以及当前最主流的基于信息检索 (information retrieval, IR) 的方法[34], 如矢量空间模型 (vector space model, VSM)[35]、潜在语义索引 (latent semantic indexing, LSI)[36]、潜在狄利克雷分布模型 (latent Dirichlet allocation, LDA)[37]、Jenson Shannon (JS) 模型[38]等. 近年来, 基于学习式的方法成为用于恢复缺失的跟踪链接的主流技术, 包括传统机器学习 (machine learning), 例如随机森林 (RF)[8,10]、朴素贝叶斯 (naive Bayes, NB)[8,39]和深度

学习 (deep learning)[13,40,41] 模型, 如循环神经网络 (recurrent neural network, RNN)、词嵌入模型 (word embedding)、基于 Transformer 的 BERT 模型[40]等. 然而大部分研究还是基于开源软件项目或学生项目[10,31,42], 如基于 CoEST 中收集的数据, 验证算法的有效性. 即使相关研究也逐渐开始采用工业界项目进行实验验证[12,14], 但是研究人员发现, 在真实的企业项目中采用当前主流的自动追踪技术构建的可追踪性模型[43], 并没有实现如以往研究在学生项目和开源项目上的理想的实验效果.

本文通过前期案例分析发现将主流的学习式的可追踪性技术应用于企业项目, 其效果不佳的原因主要是由于样本数据带来的关键挑战: (1) 数据低质量. 目前软件可追踪性主要是依赖于人工方式创建和维护, 这种手段最主要的问题是导致产生很多低质量样本, 如软件制品之间无关联、延时关联、错误关联、不完整关联等. 因此使得训练模型难以学习到正确、可靠且有价值的数据. (2) 数据稀疏性. 即使企业项目制品规模足够大, 但软件制品之间 (如缺陷-commit 之间) 有关联的跟踪链接比例 (linked ratio) 太小. 此外开源项目中适用的特征, 如过程特征, 由于企业的组织特性 (如内外研发合作模式), 部分过程特征并不适配, 这导致模型可学习的特征空间依然是稀疏的. (3) 数据不平衡性. 可追踪性模型中输入的样本是通过两两制品之间通过时间匹配规则, 生成潜在的制品跟踪对, 又因为一个制品在其生命周期内最多和 n 个 (在企业项目中 n (中位数)≤2 普遍成立) 目标制品存在真正关联, 而时间匹配规则会生成过多的无关联的跟踪对, 导致数据集中包含很多“假”链接样本, 样本数据中“真”和“假”链接样本比例过于悬殊, 因而加重了数据不平衡性.

为了克服以上挑战, 本课题组首先提出了基于半监督学习的可追踪性恢复框架 (semi-supervised pre-processing for learning-based traceability, SPLINT)[44], 简化流程如图 2 所示, 具体而言: (1) 在模型预处理阶段, 引入混合文本相似性特征, 缓解特征空间稀疏性问题; (2) 在模型训练阶段, 引入半监督学习策略, 学习无标签样本, 缓解样本规模稀疏性问题; (3) 在具体半监督样本选择策略中, 提出调整后的类平衡自训练 (adjust class balancing self-training, CBST-Adjust) 样本选择策略, 重点缓解数据不平衡问题. 通过半监督学习策略构建的可追踪性恢复框架 SPLINT 的确有效缓解了数据稀疏性和数据不平衡性问题.

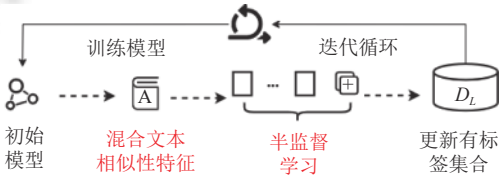


图 2 基于半监督学习的可追踪性恢复模型 SPLINT 简化流程

然而基于半监督学习的可追踪性恢复框架 SPLINT 依然存在相应的挑战. 半监督学习的主要原理是通过模型在未标签的数据上的预测情况, 来得到可靠的伪标签, 从而将大量无标签数据引入训练. 因此半监督学习有效的前提是: (1) 要生成可靠的伪标签. 然而正如上述挑战中所提到的, 由于企业组织特性的复杂性和过程交付的压力等潜在的原因, 依赖于人工 (开发人员打 tag) 方式创建和维护的软件制品间的可追踪性链接可能包含大量的低质量样本, 从而使得半监督学习模型难以有效预测得到可靠的伪标签样本. (2) 其次为了充分利用无标签样本, 半监督学习过程中要引入大量可靠的伪标签. SPLINT 动态调整无标签样本选择比例 (逐步递增), 即随着迭代, 降低 (不同类别) 的样本置信度阈值, 来引入更多的伪标签参与训练. 但是低阈值会不可避免地引入质量低的伪标签样本.

本文针对 SPLINT 中存在的问题, 进一步优化了框架, 提出了一个基于结合主动学习及半监督学习改进策略的自动化可追踪性恢复框架 (software traceability recovery framework based on active learning and semi-supervised learning), 命名为 STRACE(AL+SSL), 该框架可以针对软件可追踪性在企业数据上面临的挑战和半监督学习技术应用时的挑战, 逐一引入对应的改进策略, 从而更加有效地预测真实复杂企业项目中软件过程制品之间的可追踪性关系. 具体而言, STRACE(AL+SSL) 框架从如下几点进行改进: (1) 将主动学习与半监督学习结合, 重塑初始训练集, 让模型学习正确可靠 (reliable), 有价值且有用 (valuable and useful) 的数据, 重点改进初始训练集低质量问题,

从而提高半监督学习生成的伪标签质量; (2) 在选择无标签样本时, 引入权重机制, 结合初始训练模型精确度和伪无标签样本概率分数, 为所选择样本赋予权重值, 重点缓解半监督模型中, 伪标签数量与质量间难以达成 trade-off 导致模型效果不稳定的问题. 本文基于多个企业项目, 展开了多组对比实验, 验证了本文所提框架在当前真实企业项目软件可追踪性恢复任务上的有效性.

改进后的框架如图 3 所示. 其中左侧为主动学习阶段, 根据样本选择策略筛选无标签样本, 同时赋予样本标签. 此过程通常是邀请专家打标签, 在本文实验场景下, 为模拟真实数据集上有标签样本和无标签样本的真实分布, 将有标签集合视作全部数据集, 随机选择 30% 的样本为初始有标签样本, 剩余 70% 视作无标签样本. 在此基础上, 再进行自动化的半监督学习的伪标签生成, 选择无标签并重新训练模型.

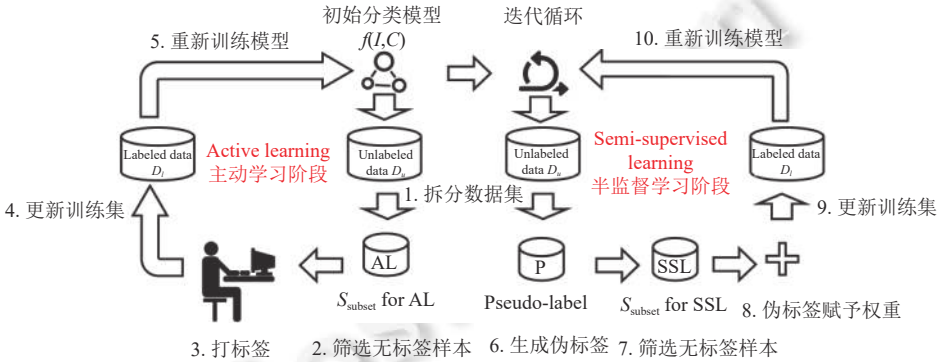


图 3 结合主动学习及半监督学习改进策略的自动化可追踪性恢复框架 STRACE(AL+SSL)

2 前期调研基础

为了解工业界场景中软件过程制品间可追踪性的现状及可追踪性模型表现不佳的原因, 本课题组与一家大型 ICT 企业合作, 在前期展开了针对软件制品间可追踪性的现状调研与实验案例探索研究, 识别分析了 3 点关键挑战.

2.1 软件可追踪性现状调研

通过调研企业软件项目中软件可追踪性质量情况, 包含需求, 缺陷, 代码和测试用例等主要制品间的关联关系 (如图 4 所示), 本文发现由于软件研发团队往往面临时间紧迫的发布压力, 因此很容易出现可追踪性低质量的问题.

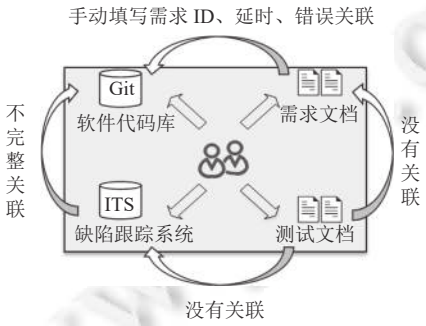


图 4 软件制品可追踪性现状

● 挑战 1. 原始数据低质量

现状 1: 软件制品可追踪性链接的可靠性难以保障. 需求可追踪性属于合作企业保障其可信软件的最佳实践之一, 为了建立可追踪性关系, 各个项目团队被要求在进行 commit 操作时手动填写功能性需求编号. 需求具体分为 3 个粒度, 系统级需求, 功能性需求和需求故事卡片 (少部分团队使用). 且版本发布前保证所有需求必须关联代

码. 这一方面反映了工业界对软件可追踪性的重视开始逐步加强. 但另一方面人为的操作依然会导致延时关联、错误关联等不可靠的质量问题. 具体而言, 延时关联的情况是由于需求关闭前紧急关联某个 commit; 错误关联也经常发生, 访谈发现一部分开发人员往往更容易将针对缺陷修复的 commit 替换为与需求进行关联. 一方面, 这是由于开发人员更熟悉各自负责的需求, 且企业对需求的可追踪性有更严格的要求. 另一方面, 开发人员大多为外包工作人员, 对组织规范性不清楚或不严格遵守, 也会导致可追踪性链接的可靠性难以保障的问题.

现状 2: 软件制品可追踪性链接的完整性难以保障. 通过分析缺陷与 commit 记录关联的比例, 本文发现企业项目中平均有超过 50% 的缺陷没有与 commit 进行关联. 这可能是由于企业更多关注需求可追踪性的影响, 而缺陷本身并没有要求与 commit 强制关联. 然而, 根据开发人员的反馈, 在每个正式发布版本中, 约 80% 的缺陷都是功能性缺陷, 需要通过 commit 对代码进行修复. 不难发现, 缺陷和 commit 之间的关联关系不完整问题尤为普遍.

现状 3: 软件制品可追踪性链接缺失. 通过分析测试用例与其他制品之间的关联关系, 本文发现即使公司要求测试人员将各自分配好的需求与对应的测试用例进行关联, 但测试用例的历史数据中, 只有少部分项目团队关联了最高级别需求 (系统级需求) 和测试用例, 仍有很多历史测试用例关联的需求编号为空值. 此外, 测试用例与缺陷之间甚至没有建立和维护关联关系.

不难发现, 各类制品之间的可追踪性缺失、延时、错误和不完整等问题都反映出企业内部的软件过程制品可追踪性质量有待改进.

2.2 案例分析

为了解可追踪性模型在企业项目上的实践效果, 本文先以 Rath 等人[10]提出的 Baseline(RF) 可追踪性模型为基础模型, 以及其实验[10]中验证过的 6 个开源项目数据为对比数据, 同时从本文合作企业选择了 10 个企业项目数据展开了探索性案例分析. 项目基本统计量信息如表 1 所示, 数据集收集截至 2020 年 10 月. 考虑到缺陷-commit 间存在一定比例的关联关系恰好可以为模型提供训练数据, 也是相关研究[10,27,31,40]实验中使用最频繁的制品, 为方便验证模型实验效果, 本文重点关注的是缺陷-commit 间的可追踪性问题.

表 1 项目数据集统计量信息

项目	类型	起始时间	缺陷	commit	总样本	“真”链接	“假”链接	无标签	有链接缺陷 (%)	有链接commit (%)	不平衡率
Derby	开源	2011-10	478	1 108	32 902	300	32 535	72 765	48.74	26.81	108
Drools	开源	2014-8	1 728	2 989	110 529	1 030	30 040	79 459	50.81	33.89	29
Groovy	开源	2014-7	1 170	5 368	342 722	1 017	223 524	118 181	69.40	18.55	220
Infinispan	开源	2015-1	2 287	5 546	268 886	1925	154 822	112 139	68.34	34.58	80
Maven	开源	2009-11	530	1 747	27 396	314	11 963	15 119	51.51	17.74	38
Pig	开源	2014-11	295	429	3 912	240	3 666	15 774	77.63	55.71	15
平均值			1 081	2 865	131 058	804	76 092	68 906	61.07	31.21	82
P1	企业	2019-2	1 764	16 912	944 236	1 157	303 421	639 658	25.06	6.84	262
P2	企业	2019-6	1 389	12 229	386 353	1 110	232 629	152 614	39.02	9.08	210
P3	企业	2019-6	448	4 888	66 754	279	29 028	37 447	28.57	5.71	104
P4	企业	2020-1	856	7 271	707 732	759	223 695	483 278	24.88	10.44	295
P5	企业	2020-3	188	1 649	20 228	257	10 147	9 824	33.51	15.59	39
P6	企业	2019-8	337	2 903	45 471	236	22 229	23 006	34.12	8.13	94
P7	企业	2019-11	405	6 114	249 120	936	149 997	98 187	45.19	15.31	160
P8	企业	2020-5	192	514	13 801	132	7 364	6 305	44.27	25.68	56
P9	企业	2019-7	270	2 437	35 621	287	20 364	14 970	45.19	11.78	71
P10	企业	2020-4	631	3 609	184 735	867	128 661	55 207	47.39	24.02	148
平均值			648	5 853	265 405	602	112 754	152 050	36.72	13.26	144

图 5 中展示了以缺陷-commit 跟踪对为样本, Rath 等人[10]提出的 Baseline(RF) 可追踪性模型在 6 个开源项目和 10 个企业项目上的实验效果. 考虑到模型结果存在随机性, 图 5 中统计了 10 次实验结果得到的平均值. 图 5 横

轴右侧表示 $F2$ 值分布, 柱状图中下半的部分表示 6 个开源项目上的模型效果分布, 上半部分表示 10 个企业项目上的 $F2$ 值分布. 不难发现, Baseline(RF) 模型在企业项目上相比于开源项目效果呈现显著下降, 平均 $F2$ 值约只有 30% 左右, 后者的平均 $F2$ 分值则接近 80%. 为进一步分析基于学习式的模型在企业项目可追踪性问题上表现不佳的潜在原因, 下面总结分析了以下关键挑战.

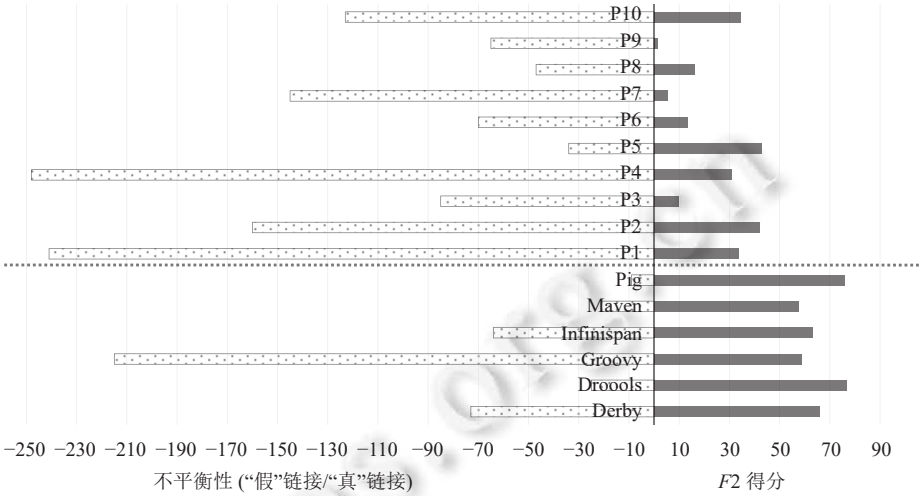


图 5 Baseline(RF) 模型在开源及企业项目上的 $F2$ 分数差异

● 挑战 2. 样本数据不平衡

首先是每个项目中样本不平衡率. 图 5 中横轴左侧表示每个项目样本不平衡性, 统计了每个项目中多类样本 (“假”链接) 和少类样本 (“真”链接) 的比例, 其计算公式如公式 (1) 所示:

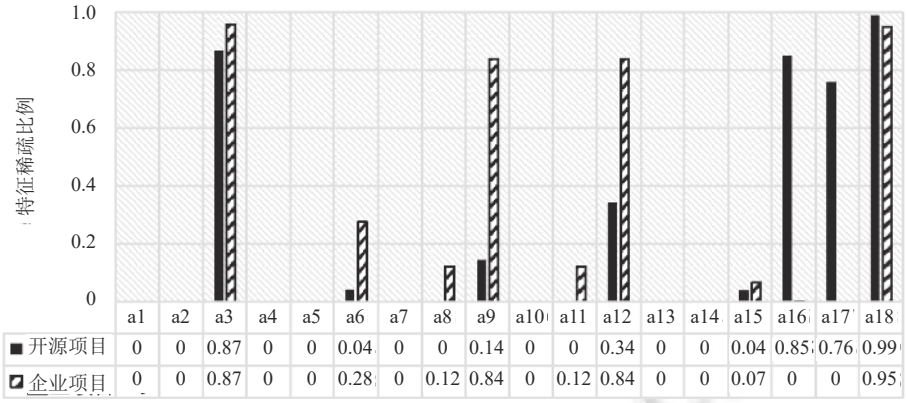
$$ImbalanceRatio = \frac{N(\text{“假”链接})}{N(\text{“真”链接})} \tag{1}$$

从表 1 中可以看到, 10 个企业项目数据集不平衡率平均高于 140. 导致企业项目样本不平衡比例高的最直观的原因在于, 有链接的缺陷和有链接的 commit 比例普遍较低, 平均来看分别只有 36.72% 和 13.26%. 具体而言, 数据不平衡性由于工业软件项目紧张的研发过程而变得愈加严重. 一方面, 企业项目以业务为导向, 研发周期紧凑, 编码频率密集. 如表 1 所示, 企业项目的源制品和目标制品规模比例存在较大的悬殊, 即大多数企业项目虽然有 $N\times100$ 量级的缺陷, 但 commit 记录则达到数千甚至数万的量级. 这是因为与开源软件项目处于维护期不同, 企业项目尚处于初期, 项目迭代较为频繁, 产生的软件制品规模较多, 尤其是一些大规模项目 commit 累积量激增, 这无疑加剧了样本数据集不平衡性的严重程度. 其次, 软件可追踪性天然地假设缺陷会被一次或多次 commit 修复. 而在企业项目中, commit 提交频率较高, 缺陷生命周期较长 (只有缺陷得到回归测试后才允许关闭), 因此更多的 commit 会被选择与缺陷生成候选跟踪对样本, 这将导致生成的样本链接标记中, 被标记为“真”的链接数量和被标记为“假”的链接数量存在极大的悬殊.

● 挑战 3. 样本数据稀疏性

从表 1 中不难发现, 由于有链接的制品占比较低, 在企业项目中尤为明显, 这导致总体样本中存在大量的无标签样本. 这无疑使得用于训练的可学习的样本量是稀疏的. 此外, 值得注意的是, Baseline(RF) 中提取的特征集在企业项目数据集上呈现了较为严重的特征空间稀疏性问题, 直接影响了采样数据集中可学习样本的信息量. 图 6 中对比了两类数据集特征矩阵中每列特征的稀疏程度, 其中横轴中 a1–a18 为 Baseline(RF) 中定义的 18 个特征, 纵轴为特征稀疏比例, 计算公式如公式 (2) 所示:

$$FeatureSparsity(\text{a}i) = \frac{N(ai\text{列空值样本数})}{N(ai\text{全部样本数})} \tag{2}$$



	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17	a18
开源项目	0	0	0.87	0	0	0.04	0	0	0.14	0	0	0.34	0	0	0.04	0.85	0.76	0.99
企业项目	0	0	0.87	0	0	0.28	0	0.12	0.84	0	0.12	0.84	0	0	0.07	0	0	0.95

图 6　特征稀疏性

与开源项目数据集相比, 企业项目中过程特征 (如 a3、a6、a9 和 a12) 呈现更高的稀疏性. 其中文本相似性特征 (如 a17) 则存在较多非空值. 因此, 虽然 Baseline(RF) 中定义的特征集 (a1–a18) 在开源软件项目中表现良好, 然而, 在选定的企业项目中, 稀疏的特征空间使得当前特征样本无法捕捉到制品间跟踪链接的细节信息. 通过访谈企业内部人员本文发现, 这极大可能是因为企业项目中记录的制品属性信息 (人、时间等字段), 存在不正确记录的数据. 即在企业项目上, 样本中学习到过程特征可能是不正确的或与事实不符的. 例如, 缺陷被分配给内部 leader 角色 (如项目经理), 而缺陷的 commit 提交人员, 即实际修复人员则可能由其他开发人员 (如外包开发者) 负责. 在大型企业组织中, 这种现象往往是一个广泛采用的合作模式.

为进一步分析选取的特征对模型的贡献, 表 2 中计算了两类数据集每个特征与标签之间的相关性, 即通过 Pearson correlation coefficient 值 (corr.) 评估特征 (a1–a18) 与标签 y (“真”=1, “假”=0) 之间的相关性. 同时对比了特征与跟踪链接标签相关性指数的显著性 p-value. 不难发现, 在企业数据集中, 与过程相关的特征 (a4–a16) 中存在显著相关 (p-value<0.05*) 的特征较少, 且与标签 y 的相关性值较低 (corr. 较小). 文本相似性 (a17, a18) 是两类数据集上都最相关的特征. 这说明了文本相似性特征的重要性, 特别是在企业项目中, 应该加强对语义信息的捕捉.

表 2　特征相关性

特征	开源项目		企业项目	
	相关性 (corr.)	显著性 (p-value)	相关性 (corr.)	显著性 (p-value)
a3	0.1	0.0742	0.2	0.0188*
a4	−0.11	0.0229*	−0.06	0.0296*
a5	−0.12	0.0367*	0.01	0.1682
a6	0.02	0.0257*	0.05	0.0238*
a7	−0.13	0.0318*	−0.01	0.1706
a8	−0.06	0.0705	0	0.4963
a9	0.11	0.0300*	0	0.3710
a11	−0.12	0.0048**	0	0.4903
a12	0.1	0.1158	0	0.3549
a14	−0.06	0.0725	0	0.2003
a15	−0.02	0.191	0.01	0.1795
a16	−0.08	0.0309*	−0.06	0.0267*
a17	0.41	0.0000**	0.36	0.0457*
a18	0.27	0.0006**	0.31	0.0240*

注: *表示0.05显著水平, **表示0.01显著水平

3　基于主动学习-半监督学习的可追踪性恢复框架 STRACE(AL+SSL)

可追踪性算法主要是通过制品之间潜在的关联线索判断分类标签是否为 (“真”=1, “假”=0) 链接的二分类模

型. 例如, 为了将源制品 (例如需求、缺陷) 与目标制品 (例如代码文件、代码提交) 联系起来, 可追踪性模型学习源制品和目标制品跟踪对之间的多维度特征, 如文本相似性、时间关系、人物关系等, 预测最相关的跟踪对作为追踪链接. 下面会简要介绍 STRACE(AL+SSL) 可追踪性恢复框架的构建流程, 本节将其分为数据预处理阶段 (参考了 Rath 等人[10]提出的 Baseline(RF) 模型构建流程) 和后续的训练阶段, 并重点在训练阶段详细介绍主动学习结合半监督学习的具体改进策略.

3.1 数据预处理阶段

可追踪性恢复框架 STRACE(AL+SSL) 包含 5 个主要的预处理步骤: 样本生成, 样本标注, 特征提取, 样本分割和样本重采样.

3.1.1 样本生成

为生成数据样本, 需要从不同数据库中抽取需要构建其可追踪性的软件制品, 相互匹配生成两两制品间的候选跟踪对. 本文从合作企业中申请到了几类制品的软件资源库访问权限. 首先从制品数据库表中, 选择明确“已修复 (resolved)”且“已关闭 (closed)”的缺陷 (issue, I), 和“已合入代码仓 (merged)”的代码提交记录 (commit, C), 本文分别将上述两类制品视为源制品和目标制品. 参考 Rath 等人[10]提出的基于时间规则的跟踪对匹配函数 $is\_candidate$ 生成每个项目中的候选跟踪对样本.

$$is\_candidate(I,C) = created(I) \leqslant committed(C) \leqslant resolved(I) + \varepsilon \tag{3}$$

$$\varepsilon = median\_per\_project(|committedTime(C) - resolvedTime(I)|) \tag{4}$$

值得注意的是, 本文调整了 $is\_candidate$ 函数中参数 ε, 即延迟间隔参数. 在以往研究中[41,45], ε 往往设为常数, 如小于等于 7 天. Rath 等人[10]选择将其设为一个静态变量, 具体而言他们统计了 6 个开源项目上相互关联的缺陷-commit 的代码提交时间在缺陷关闭时间后的间隔时间绝对值分布后, 将 ε 设为 30 小时, 然后将 $\varepsilon = 30$ 应用在公式 (3) 中. $is\_candidate$ 函数中 ε 的设置一方面是确保跟踪对匹配函数不会完全丢弃缺陷关闭后再提交的 commit, 另一方面其上限值保证 $is\_candidate$ 函数不会无限制地匹配缺陷和之后提交的所有 commit 记录. 尽管企业规则是不允许缺陷“已关闭”或“已解决”后还会存在 commit 与之关联且被提交, 但从历史数据的统计情况来看, 这的确是普遍存在的现象. 此外考虑到本文分析的企业项目涉及多种不同业务领域、不同规模及不同团队组成的项目数据, 统一的静态变量值 $\varepsilon\varepsilon$ 可能并不适用于所有企业项目. 因此本框架中通过公式 (4) 计算了每个项目的 ε, 从而在样本选择和生成环节, 使得 $is\_candidate$ 函数匹配每个项目的特性, 以确保生成的数据集中不遗漏过多的“真”链接样本.

3.1.2 样本标注

对每条跟踪对进行标注, 若 commit 的附属信息字段中存在开发人员标注的某个缺陷的 ID, 则将其对应的跟踪对标签标注为“真”链接, 否则标注为“假”链接. 值得注意的是, 基于第 2 节的前期调研及表 1 中的历史数据可以发现在大部分项目中, 相当大比例的缺陷及 commit 制品存在没有任何关联链接的情况, 则此部分制品生成的跟踪对被视为无标签样本 (标注为“NULL”). 表 1 给出了 16 个项目上的总样本数, “真”和“假”链接样本数和无标签样本数.

3.1.3 特征提取

在本文第 2 节中讨论了 Rath 等人[10]提出的 18 个过程维度特征 (a1–a18, 如表 3 所示), 如人物关系、制品时间关系等特征在企业数据集上的特征稀疏性问题. 主要原因是由于企业组织特性, 链接样本的过程特征可能是不正确的或与事实不符的. 例如, 缺陷历史数据中被指定负责人员往往是企业内部工程师, 如承担项目 leader 角色的项目经理等, 但实际的 commit 提交者, 即实际的缺陷修复者通常是具体的开发人员 (大多数情况下为外包开发人员). 内外研合作的团队组织形式 (每个项目团队内约 20% 的企业内研工程师, 80% 的外包员工) 在大型企业组织中是广泛采用的合作模式. 此外由于内研工程师负责多种任务, 且缺陷的关闭需要确保对应修复的代码完成自测和回归测试等流程, 导致缺陷的修复或关闭时间往往不能被及时记录. 由此在企业项目中 Rath 等人提出的过程特征无法提供正确有效的信息量, 从而造成特征空间稀疏的问题, 进一步导致了模型表现不佳的结果.

表 3 特征集合

特征	ID	描述
人物特征	a1	$developer(I)$
	a2	$committer(C)$
	a3	1, (if $committer(C) \in developer(I)$); 0, otherwise
制品时序特征	a4	$committed(C) - created(I)$
	a5	$resolved(I) - committed(C)$
	a6	1 (if a4 and a5 $\geqslant$ 0); 0 otherwise
	a7	$\lvert a5\rvert < \varepsilon, \varepsilon = median\_perproject(\lvert a5\rvert)$
前序关联制品特征	a8	$committed(C) - committed(Cp)$, $Cp = \{\max Committed(Cx) \mid is\_linked(Cx,F) \wedge committed(Cx) < committed(C)\}$
	a9	$overlap(C,Cp), overlap = (\mathrm{mod}(C) \cap \mathrm{mod}(Cp)) / \max(\mathrm{mod}(C), \mathrm{mod}(Cp))$
	a10	$committer(Cp)$
后序关联制品特征	a11	$committed(Cn) - committed(C)$, $Cn = \{\min Committed(Cx) \mid is\_linked(Cx,I) \wedge committed(Cx) > committed(C)\}$
	a12	$overlap(C,Cn), overlap = (\mathrm{mod}(C) \cap \mathrm{mod}(Cn)) / \max(\mathrm{mod}(C), \mathrm{mod}(Cn))$
	a13	$committer(Cn)$
当前潜在匹配制品特征	a14	$I\_opened = \lvert\{Ix \mid opened(I) \wedge created(I) \leqslant committed(C) \leqslant resolved(I)\}\rvert$
	a15	$\lvert developer(I\_opened)\rvert$
	a16	$C\_linked = \lvert\{Cx \mid is\_linked(Cx,I) \wedge committed(Cx) < committed(C)\}\rvert$
制品多维文本相似性特征	a17	$VSM(sim(description(I), message(C))$
	a18	$VSM\text{-}2gram(sim(description(I), message(C))$
	a19	$LSI(sim(description(I), message(C))$
	a20	$JS(sim(description(I), message(C))$
	a21	$LDA(sim(description(I), message(C))$
	a22	$NMF(sim(description(I), message(C))$
制品混合文本相似性特征	a23	$VSM + LDA(sim(description(I), message(C))$
	a24	$JS + LDA(sim(description(I), message(C))$
	a25	$VSM + NMF(sim(description(I), message(C))$
	a26	$JS + NMF(sim(description(I), message(C))$
	a27	$VSM + JS(sim(description(I), message(C))$

但通过观察历史制品数据, 本文发现企业对于需求、缺陷和测试用例等制品的文本描述比较丰富, 一些团队还会制定描述规则及模板. 因此本文通过充分挖掘和利用企业项目中制品间的语义信息, 帮助模型学习更加有效的特征, 从而区分“真”和“假”链接. 需求补充的是, 企业项目由于数据保密性规则 (源代码保密), 将 Rath 等人提出的缺陷文本描述与代码提交文件的文本相似性特征舍去, 计算了以下多维度混合文本相似性特征.

● 多维文本相似性 (a17–a22). 本文使用了 6 种主流的 IR 模型度量了制品文本相似性, 包括 *VSM*, *VSM-2gram*[46], *LSI*, *JS*, *LDA* 和 *NMF*[12].

● 混合文本相似性 (a23–a27). 考虑到单一文本相似性特征稀疏性问题比较严重, 本文参考了 Gethers 等人[47] 的工作, 使用混合文本相似性特征, 通过归一化加权, 结合不同 IR 模型度量制品文本相似性特征.

3.1.4 样本分割

如图 7 所示, 本文将标注的样本数据依据时间顺序进行分割, 参照 Rath 等人选择的分割参数, 比例为 4:1, 将标注数据分为 80% 的训练集和 20% 的测试集. 选择时序分割法是因为考虑到制品之间存在时序关系, 其次特征集合中包含时序特征 (a4–a7), 且训练集样本在测试集样本之前更加符合真实可追踪任务中实践场景[10,48], 即通过历史数据预测未来的制品链接.

3.1.5 样本重采样

不难发现, 通过样本跟踪对匹配函数 *is_candidate* 生成的样本是严重不平衡的, 其中包含了过多“假”链接样

本. 使用极其不平衡的数据进行训练, 会使得分类模型倾向于学习多类样本, 实验结果会更容易将跟踪对都预测为“假”链接. 为了缓解样本中不平衡比例的影响, 需要对数据进行平衡. 参考当前的可追踪研究[8,49], SMOTE 为重采样方法中表现最优的一种算法, 故本文选择了 SMOTE 算法对训练集 train_set 重采样, 再使用不平衡测试集 test_set 预测跟踪对的结果, 评估分类器效果. 值得注意的是, 在后续训练阶段引入改进策略 (如主动学习结合半监督学习步骤) 后, 每轮迭代会选择新的无标签样本加入原始训练集, 训练集再次按照时序排序后, 样本重采样步骤会被重新启动, 即对更新的训练集 new_train_set 使用再进行 SMOTE 重采样, 用于缓解每次迭代中训练集的不平衡性.

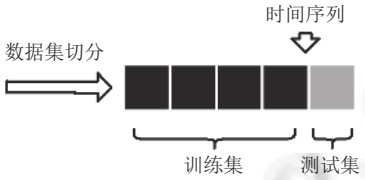


图 7 样本时序分割

3.2 训练阶段

针对第 2 节中讨论的几点挑战, STRACE(AL+SSL) 框架重点通过训练阶段引入不同的策略, 针对性地改进训练数据特性. 具体改进策略有以下几点.

● 引入半监督学习 (semi-supervised learning, SSL) 策略, 旨在通过学习大量可靠的无标签样本, 改进样本稀疏性问题. 同时在半监督样本选择机制中, 提出调整后的 CBST-Adjust 选择策略, 重点缓解数据不平衡问题.

● 引入主动学习 (active learning, AL) 策略, 重塑初始训练集, 同时与半监督学习策略相结合, 重点减轻初始训练集低质量的影响, 提高半监督学习生成的伪标签质量.

● 引入权重机制, 选择无标签样本时, 考虑初始训练模型精度和伪标签的可靠性的影响, 为不同样本赋予不同权重, 重点缓解半监督模型由于引入低质量伪标签而导致的效果不稳定和无标签利用率不高的问题.

3.2.1 软件可追踪性半监督学习策略

半监督学习的主要原理是根据模型在无标签样本上的预测情况, 来得到伪标签, 再将大量可靠的无标签数据引入训练. 在以往的半监督学习研究中, “置信度阈值 (confidence threshold)”是一种比较主流的伪标签选择方式. 比如在 FixMatch[50]中, 置信度≥阈值 (0.9) 的伪标签样本会直接加到初始训练集中. 通过设定较高的阈值, 伪标签的质量 (即正确性) 可以得到保证. 但是, 在无标签样本存在严重的样本不平衡的场景下, 统一的伪标签阈值会使得半监督学习在每轮迭代选择的无标签样本依然偏向多类样本, 即本文中的“假”链接样本.

(1) 类平衡样本选择策略

故本文参考了 Zou 等人[51,52]提出的 CBST (class balancing self-training) 策略和 Wei 等人[53]提出的 CReST (class-rebalancing self-training) 策略, 及 Chen 等人[54]提出的 SimiS (simple yet overlooked Baseline(RF) for imbalanced semi-supervised learning) 策略.

● CBST 样本选择策略

CBST 策略基于初始训练模型, 得到伪标签样本在每个类别上的预测概率值 probability score (多类, 少类). 为了选择一定比例的高可靠性的伪标签样本, 与以往研究中统一设置置信度阈值 (如 0.9) 不同的是, CBST 主张按照类别, 为每个类别独立设置选择无标签样本的置信度阈值. 从而可以缓解由于设置统一的置信度阈值而导致的样本不平衡问题. 同时, 与直接设定置信度阈值不同的是, CBST 从每类样本选择比例为 $top(p)$ 的伪标签, 此时每类样本的置信度阈值因不同样本的概率值 probability score (多类, 少类) 分布而不同. 这样可以避免由于设置过高的常量阈值, 使得每轮迭代很难获得需要被学习到的少类样本的问题. 以少类样本 (即“真”链接样本) 为例, 选择“真”链接样本的置信度阈值计算如公式 (5) 所示:

$$e^{T_1} = ProbList[top(p) \times D_{\text{unlabeled},1}] \tag{5}$$

其中, p 为少类样本选择比例, $D_{\text{unlabeled},1}$ 为将伪标签预测为“真”链接的无标签样本集合, *ProbList* 为对 *Probability-Score* 从大到小排序的概率值列表. 确定每类样本的选择比例 *top*(p) 和置信度阈值后, 少类样本每轮迭代选出的伪标签集合 $S_{\text{unlabeled},1}$ 如公式 (6) 所示:

$$\text{if } ProbabilityScore(i) \geqslant e^{T_1}, \text{ then } i \in S_{\text{unlabeled},1} \tag{6}$$

● CReST 样本选择策略

CReST 策略主张采用不平衡的样本采样比例, 来设置伪标签的筛选规则. 即通过计算有标签样本集合的不平衡性, 制定每类样本伪标签选择比例 μ, 确保半监督环节选择机制更偏向少类样本. 在可追踪性场景下, CReST 样本选择策略具体计算公式如公式 (7) 和公式 (8) 所示:

$$\mu_0(\text{多类伪标签采样比例}) = \left(\frac{N(\text{少类“真”链接 in } D_{\text{labeled}})}{N(\text{多类“假”链接 in } D_{\text{labeled}})}\right)^{\alpha} = (1/ImbalanceRatio(D_{\text{labeled}}))^{\alpha}, \text{ where } \alpha \geqslant 0 \tag{7}$$

$$\mu_1(\text{少类伪标签采样比例}) = \left(\frac{N(\text{少类“真”链接 in } D_{\text{labeled}})}{N(\text{少类“真”链接 in } D_{\text{labeled}})}\right)^{\alpha} = 100\%, \text{where } \alpha \geqslant 0 \tag{8}$$

● SimiS 样本选择策略

SimiS 策略主张使用更加简单激进的方法, 仅仅用尽可能多的少数类伪标签扩展已标注集合以构造平衡的训练集, 而不再增加最多数类的伪标签样本. SimiS 样本选择策略的具体计算公式如公式 (9) 和公式 (10) 所示:

$$S_0 = \beta\left(\frac{N(\text{多类“假”链接 in } D_{\text{labeled}}) - N(\text{多类“假”链接 in } D_{\text{labeled}})}{N(D_{\text{labeled}})}\right) \tag{9}$$

$$S_1 = \beta\left(\frac{N(\text{多类“假” 链接 in } D_{\text{labeled}}) - N(\text{少类“假” 链接 in } D_{\text{labeled}})}{N(D_{\text{labeled}})}\right) \tag{10}$$

其中, β 是控制不平衡性的超参数. 理想情况下, 设置 $\beta = N(D_{\text{labeled}})$ 可以使得初始有标签集合完全在各类别上分布完全均衡.

● CBST-Adjust 样本选择策略

实际上, 上述几种从选择伪标签样本思想都致力于平衡新训练集中的样本比例. 核心区别在于通过不同的策略为每个类添加的伪标签的数量. CBST 通过分类别设置阈值, 确保少类也能够被选中, 但由于无标签样本也是极其不平衡的, 多类伪标签的绝对数量依然较多, 因此会进一步加重训练集的不平衡性. CReST 通过设置不平衡采样比例控制多类伪标签, 虽然这在很大程度上增加了少类的数据数量 (少类伪标签的采样比例为 100%), 但不可避免地引入了质量较低的少类的伪标签, 同时也为多类增加了许多样本. SimiS 通过设置更加简单激进的平衡超参数 β, 直接舍去多类伪标签, 且通过选择少类伪标签, 达到尽可能均衡训练集类别分布的目的, 但在该策略中, 一种常见的情况是所需的少类伪标签的数量甚至可能超过未标记集合中少类的总量, 因此很难为 SimiS 策略的平衡超参数 β 设置合理的取值.

通过分析上述 3 种半监督类平衡样本选择策略的优缺点, 本文提出了改进后的 CBST-Adjust 样本选择策略. 如图 8 所示, 该策略按照置信度从高到低排序伪标签样本, 选择每个类别中 *top*(p) 的伪标签, 确保选择高质量的少类伪标签, 再参考 CReST 策略中的不平衡采样比例 μ, 进一步限制多类伪标签的数量, 重点改进半监督环节所选择无标签样本中的多类和少类伪标签数量过于悬殊的问题.

固定的阈值以及过高的阈值都可能会丢弃很多不确定的伪标签 (尤其在迭代后期), 导致伪标签“利用率低”. 考虑到后续迭代会存在很多低阈值样本, 同时为了确保每轮迭代都能够选择到一定比例的无标签样本, 尤其是少类样本. 本文参考一系列动态阈值的工作[55,56], 也引入了随迭代 *iterationNum* 增大而降低阈值的机制. 通过参数 *p*/*increaseRate* 组合动态改变阈值, 其中 p 为初始迭代样本选择比例, *increaseRate* 为每轮迭代样本选择比例增长比例. 具体的阈值计算公式如公式 (11) 所示:

$$p_{iNum} = p_{\text{initial}} + increaseRate \times iterationNum \tag{11}$$

因此, CBST-Adjust 每轮迭代选择的伪标签集合如公式 (12) 所示:

$$S_{\text{subset}} = (\mu_0 \times top(p_{iNum}) \times D_{\text{unlabeled},0}) \cup (\mu_1 \times top(p_{iNum}) \times D_{\text{unlabeled},1}) \tag{12}$$

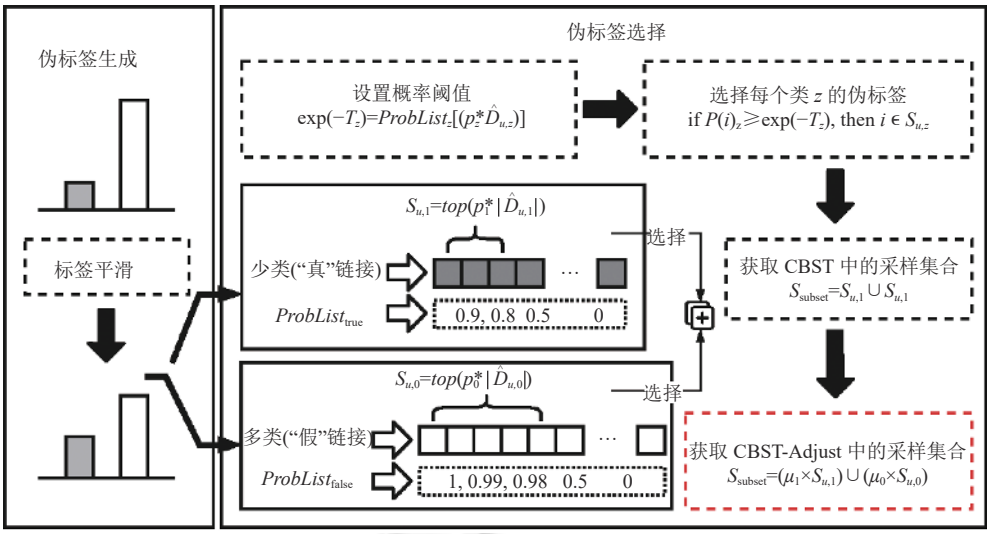


图 8 半监督学习 CBST-Adjust 样本选择策略

(2) 软件可追踪性半监督学习策略问题解析

虽然半监督学习通过学习大量无标签样本缓解了样本稀疏性, 但本文通过在可追踪性数据集 (企业数据集) 上进行初步探索后发现, 这类策略仍然存在一个明显的问题: 使用当前有标签样本进行训练, 由于初始模型效果并不理想, 生成的伪标签可能预测错误, 从而影响模型训练效果的提升.

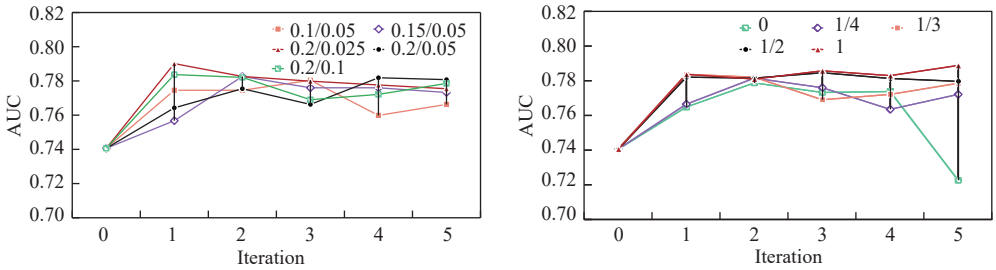
本文在表 4 中比较了可追踪性初始模型使用半监督学习 (参数设置为: α=1/3, 初始迭代样本选择比例及增长比例为 $p/increaseRate$ (0.2/0.1), 迭代范围为 0–5 次) 后的模型效果. 实验结果表明, 初始模型效果的确对半监督算法的提升程度有较大的影响. 例如, P7 和 P8 项目的 $F2$ 值提升程度小于 10%. 在半监督学习中, 由于选取的无标签样本生成的伪标签可能是不可靠的, 导致半监督模型效果在初始模型基础上提升的空间非常有限.

表 4 半监督模型提升率 ($F2$ 值) (%)

方法	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
初始模型	28.58	31.32	15.32	30.35	33.43	28.39	8.99	15.17	9.27	24.10
半监督模型	46.48	45.36	37.50	43.08	48.44	47.30	13.05	21.43	26.01	36.10
Improvement	17.90	14.04	22.18	12.73	15.01	18.91	4.06	6.26	16.74	12.00

其次, 半监督模型效果起初由于训练学习样本的增加, 尤其是少类样本的增加有明显的提升. 如图 9 所示, 本文比较了在 P5 项目上使用半监督学习策略后不同参数设置对于模型效果的影响. 图 9 的参数设置范围为, $p \in (0.1, 0.2)$, $increaseRate \in (0.025, 0.05, 1)$, $\alpha \in (0, 1/4, 1/3, 1/2, 1)$. p 和 $increaseRate$ 值越高表示在每轮迭代选择的无标签样本量更多, 但随着迭代次数的增加, 剩余的无标签样本中少类样本有限, 极易引入低质量的样本. 导致在后续迭代中不升反降的趋势愈加明显. 从公式 (7) 和公式 (8), 可以推测出, α 约接近于 0, 则多类采样比例 μ_0 值越高, 则选择的伪标签样本不平衡性越严重.

本文提出的 STRACE(AL+SSL) 框架针对半监督模型的痛点引入了两项改进策略, 一方面通过主动学习策略重塑训练集, 提高训练集样本的准确性和代表性. 另一方面, 引入权重设置机制, 选择伪标签时考虑初始训练模型精度和伪标签样本可靠性的影响, 分别赋予不同权重. 从而减少低质量伪标签样本中的噪声对可追踪分类器的影响.



(a) 伪标签初始迭代样本选择比例及增长比例 *p*/*increaseRate* 参数分析 (α=1/3)

(b) 伪标签每类样本选择比例指数 α 参数分析 (*p*/*increaseRate*=0.2/0.1)

图 9 半监督模型参数影响分析

3.2.2 主动学习策略

考虑到昂贵的成本限制, 如何寻找最有用且最正确的链接样本来构建训练样本是提升模型最关键的问题之一. 本文拟使用主动学习[57,58]来重塑软件可追踪性训练样本, 其主要原理是选择高信息量的样本进行标注(如请专家标注), 提高模型的性能. 与半监督学习中生成伪标签不同的是, 主动学习过程中的标注结果往往被认为是准确的, 从而确保重塑训练样本的可靠性. 此外主动学习策略中关注的是如何设计样本选择策略, 以提升标注样本的价值, 也就是说, 这些样本是否值得被专家标注, 以确保重塑的训练样本具备足够的信息量和代表性.

在以往的主动学习工作中[57−60], 主要以“不确定 (uncertainty sampling)”策略为主, 学习器的预测结果越不确定的样本, 信息量越多. 因此该查询策略通常通过一些方法计算样本的不确定性, 判断其是否出于决策边界, 是否携带有足够的信息量. 例如, Lewis 等人[61]认为二分类中样本的类别预测概率越接近 0.5, 其不确定性越高. 而 Scheffer 等人[62]则计算样本的熵值 (entropy) 来判定其携带信息量的多少, 熵越高的样本, 其携带的信息量越高. 上述主动学习方法的简要流程如图 10 所示, 这类方法依赖于训练集得到初始模型, 计算无标签样本的不确定性值, 选择不确定性值最高的无标签样本给专家标注.

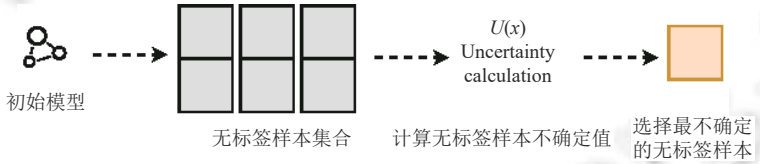


图 10 基于不确定性的主动学习简要流程

在本文引入主动学习策略时, 既要考虑到可追踪性任务面临的数据稀疏性和不平衡性等各项数据挑战对模型带来的影响, 又要确保在实践中专家标注的成本是尽可能小的. 因此参考了 Kothawade 等人[63,64]为解决主动学习中的类不平衡问题, 所提出的子模块互信息 (submodular mutual information, SMI) 函数[65,66]. Kothawade 等人[67]主张主动学习策略不仅要选择信息量最大和最多样化的样本, 还要选择与特定目标 (即 target 子模块, 在可追踪性问题中为数据集中的“真” 链接集合) 相似的样本. 基于子模块互信息的主动学习采样策略使用 SMI 函数评估待标注集合的标注价值, 使用该策略进行采样的流程如图 11 所示. 相关实验表明[67], Flqmi 采样函数相对于其他 SMI 采样函数可以在类不平衡、样本冗余的场景中实现更好的分类性能, 本文考虑到可追踪性任务也存在类似的问题, 因此在主动学习样本选择策略实现了 Flqmi, 其函数表达式如公式 (13) 所示:

$$\text{Flqmi}(S,Q)=\sum_{i\in Q}\max_{j\in S}S_{ij}+\sum_{i\in S}\max_{j\in Q}S_{ij} \tag{13}$$

其中, S_{ij} 为样本 i 与样本 j 的特征相似度, 本文参考 Kothawade 等人[67]的设置选择了余弦相似度. $S\in D_{\text{unlabeled}}$ 为拟选择的样本集合 (select 模块), $Q\in D_{\text{unlabeled}}$ 为已标注的少类“真”链接样本集合 (target 模块). 公式 (13) 中的第 1

项对于 Q 中的每个“真”链接样本选择 S 中最相似的样本并将相似度加和, 第 2 项对于 S 中的每个“真”链接样本选择 Q 中最相似样的本并将相似度加和, 两项相加同时度量了集合 S 偏向少类程度和特征空间覆盖程度. 故最大化 SMI 函数可以帮助选择既偏向于少类样本、又尽可能覆盖特征空间的待标注样本集合.

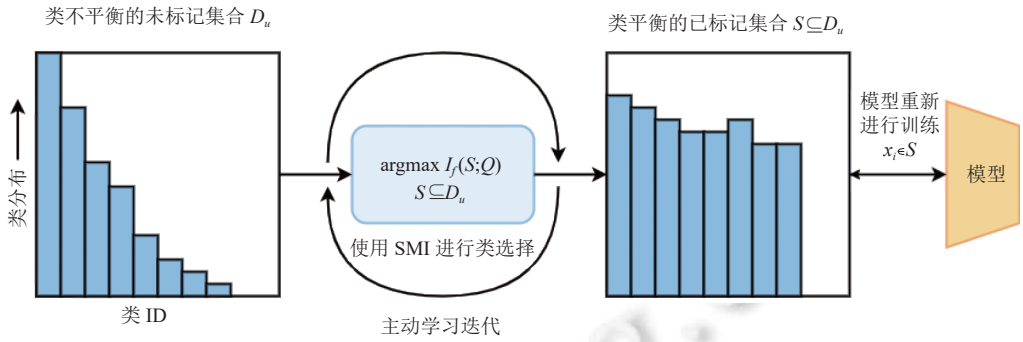


图 11 基于 SMI 的主动学习流程

3.2.3 权重机制

半监督学习另一个问题在于随着迭代次数增长, 如何平衡 (trade-off) 所选择的无标签样本“数量-质量”问题, 使得模型学习到大量的高质量新样本. 本文参考 Chen 等人[68]提出的 SoftMatch 模型, 为无标签样本设置权重参数. Chen 等人指出半监督学习中过高的阈值会导致无标签样本利用率低, 即使假设所利用的伪标签大部分是正确的 (高质量), 但仍然无法学习到好的分类模型. 然而, 如果训练过程中设置较低的阈值以提高利用率 (例如使用 CBST-Adjust 动态降低阈值), 那么伪标签中会引入过多的错误标签. 考虑到以往的半监督模型对于样本权重“缺乏合理设计”, 所选择策略都将样本权重假定为均匀分布, SoftMatch 通过对于置信度较高的无标签样本, 设置更高的权重, 对于置信度较低的更容易出错的无标签样本, 设置更低的权重, 在充分利用无标签样本信息的同时, 降低伪标签噪声的影响.

无标签样本估算权重 $\lambda(p)$ 的具体定义如公式 (14) 所示:

$$\lambda(p)=\begin{cases}\exp\left(-\dfrac{(\max(p)-\mu_t)^2}{2\sigma_t^2}\right), & \text{if } \max(p)<\mu_t\\ 1, & \text{otherwise}\end{cases} \tag{14}$$

SoftMatch 中无标签样本权重被假设服从高斯分布 $N(\mu_t,\sigma_t)\mu_t$ 的左侧部分, 并且将置信度超过均值 μ_t 的高置信度样本的权重设置为 1, 公式中的 μ_t 和 σ_t 可以由无标签样本的置信度分布估算而得.

值得注意的是, 考虑到半监督初始模型效果对生成的伪标签质量的影响, 本文在 SoftMatch 对无标签设置的权重基础上, 附加了一层权重设置. 即将初始训练模型精度设为权重 $\lambda_{\text{precision\_initialModel}}$, 故无标签样本最终权重值为公式 (15) 所示:

$$\lambda(\text{无标签})=\lambda_{\text{precision\_initialModel}}\times\lambda(p) \tag{15}$$

4 实验分析

4.1 实验数据

本文在合作企业的 10 个项目的缺陷-commit 跟踪对样本上进行实验. 这部分项目的业务类型包含两大类, 应用程序类 (P1–P4) 和服务平台类 (P5–P10) 项目. 应用程序类项目主要面向 toC 用户, 由于交付节奏快、发布压力大、更多的工程师参与项目研发, 导致项目规模较大、制品迭代频率较高, 因此缺陷的可追踪性比例较低, 后续维护成本也高. 相比之下, 服务平台类项目主要面向 toB 用户, 迭代周期相对较长, 制品数量规模较为可控, 故制品可追踪性的构建和维护相对规范. 表 1 给出了数据集所对应的详细统计量信息. 脱敏后的样本特征数据集已共享.

4.2 评价指标及基准模型

本文使用可追踪性模型中常用的指标 *Precision*, *Recall*, *F*2 及 AUC. 这几个评价指标也广泛用于之前的软件可追踪性研究工作中[10,40,44].

$$Precision = \frac{TP}{TP+FP} \tag{16}$$

$$Recall = \frac{TP}{TP+FN} \tag{17}$$

在可追踪性研究中, 不同任务场景关注的指标不同, 在恢复软件制品间的缺失跟踪链接的任务中, 更加期望预测结果能够得到较高的召回率[40], 因此本文选择 *F*2 用来衡量分类器的综合性能, 作为精度和召回率的权衡.

$$F2 = \frac{5\times Precision\times Recall}{4\times Precision+Recall} \tag{18}$$

此外, AUC 值是模型 ROC 曲线下的总体面积, 该曲线在二维空间中绘制, 以假阳性率 (false positive rate) 作为 x 坐标, 真实阳性率 (true positive rate) 作为 y 坐标. 由于 AUC 不受类别不平衡的影响并且独立于预测阈值, 因此被广泛使用.

通过与传统的基于信息检索的方法和传统的基于学习式的监督方法进行比较, 可以验证本文所提出的基于主动学习及半监督的框架 STRACE(AL+SSL), 在真实的企业项目的软件可追踪性恢复任务中的有效性. 具体而言, 本文选择了主流的基于信息检索的矢量空间模型 IR(VSM) 及基于传统机器学习的随机森林模型 Baseline(RF)[10] 进行了比较. 同时, 本文还与完全基于半监督的可追踪性框架 SPLINT[44]进行了比较. 为避免不同分类算法的影响, 本文 STRACE(AL+SSL) 框架及半监督 SPLINT 框架的基础模型都使用了 Baseline(RF) 中的随机森林算法.

4.3 实验研究问题

为评估基于主动学习及半监督学习的可追踪性恢复框架 STRACE(AL+SSL) 的有效性, 本文回答了以下研究问题.

- RQ1: STRACE(AL+SSL) 是否可以获得比其他主流的可追踪性模型更好的结果?
- RQ2: STRACE(AL+SSL) 框架中各阶段 (半监督及主动学习) 最佳的样本选择策略是哪种?
- RQ3: STRACE(AL+SSL) 框架中各模块 (半监督、主动学习、权重机制) 的贡献是怎样的?

4.4 实验参数配置

在本文的实验中, 为了方便与 Rath 等人[10]进行比较, 随机森林算法被选择作为各模型的分类算法. 实验环境配置是使用 OpenStack Nova 服务器, 英特尔酷睿处理器 (Skylake) 和 4 GB 内存. STRACE(AL+SSL) 框架中半监督 CBST-Adjust 策略的参数设置为 *p*/*increaseRate*=0.2/0.1, α=1. 主动学习策略选择 1% 数据集大小作为迭代步长进行标注. 整个实验迭代次数为 20 次. 在实验结果中汇报的值是多个项目进行 10 次重复实验后的平均值. 其中, 每次实验的结果是在一致的参数配置下统计得到的最佳 *F*2 值对应的模型结果. 为了模拟真实数据集上有标签样本和无标签样本的真实分布, 本文将 100% 的有标签集合视作每个项目的全部数据集, 并从中随机选择 30% 的样本为初始有标签样本, 剩余 70% 视作无标签样本. 30% 的取值是参考了表 1 中展示的真实企业项目中有链接的缺陷比例的平均值.

4.5 实验结果与分析

RQ1: STRACE(AL+SSL) 是否可以获得比其他主流的可追踪性模型更好的结果?

为了验证这个问题的结果, 本文基于 STRACE(AL+SSL) 框架构建的可追踪性分类模型与传统主流模型, 如基于信息检索矢量空间模型 IR(VSM), 基于随机森林的机器学习模型 Baseline(RF) 方法, 及本课题组在前期 (发表在 ESEC/FSE2022) 工作[44]中提出的基于半监督学习的模型 SPLINT 进行了对比实验, 实验的结果如表 5 所示. 为方便验证本文提出框架 STRACE 的通用性, 表 5 中也展示了 Baseline(RF)[10] 验证过的 6 个开源项目上的实验结果. 表 5 中粗体数值为众多项目在每个实验指标上最高值, Avg.行中的下划线数值为每类项目在每个实验指标

上的平均值, Imp.行中的下划线数值为 STRACE(AL+SSL) 与其他基准方法性能提升绝对值比例. *p*-value<0.05*表示显著提升, *p*-value<0.01**表示高度显著提升.

表 5 STRACE(AL+SSL) 与基准方法性能比较 (%)

项目	IR(VSM)				Baseline(RF)				SPLINT				STRACE(AL+SSL)			
	Precision	*Recall*	*F*2	AUC	*Precision*	*Recall*	*F*2	AUC	*Precision*	*Recall*	*F*2	AUC	*Precision*	*Recall*	*F*2	AUC
Derby	21.5	59.2	35.1	77.8	39	40	39.8	69.6	47.4	67.5	61	83.2	**57.1**	**77.5**	**70.7**	**88.3**
Drools	60.3	59.7	47.9	79.1	95.7	47.3	52.6	73.6	**100**	47.7	53.1	73.8	95.9	**78.5**	**79.8**	**89**
Groovy	36.3	29	24.2	64.3	86.6	73.2	75.5	86.6	**87.9**	80.8	81.4	90.3	80.5	**91.1**	**87.8**	**95.5**
Infinispan	42.6	41.3	33.3	70.2	76.8	60.8	63.4	80.3	**85.9**	62	65.4	80.9	74.6	**79.8**	**78.3**	**89.7**
Maven	20	39.1	26.3	68.3	66.7	23.5	27	61.6	66.1	22.6	25.9	61.1	**79.1**	**76.5**	**76.8**	**87.9**
Pig	70.6	74.2	58.8	84.1	87.9	93.6	92.4	96.2	94.7	95.7	95.5	97.6	**96.9**	**100**	**99.4**	**99.9**
Avg.	41.9	50.4	37.6	74	75.4	56.4	58.5	78	80.3	62.7	63.7	81.2	**80.7**	**83.9**	**82.1**	**91.7**
Imp.	38.8	33.5	44.5	17.7	5.3	27.5	23.6	13.7	0.4	21.2	18.4	10.5	—	—	—	—
P1	**51.1**	14.3	13.4	57.1	21.6	55.2	33.7	77.3	29	59.9	49.2	79.7	45.9	**72**	**64.1**	**85.8**
P2	62.4	40.5	34.8	70.2	53.7	52.1	41.9	75.9	**87.4**	70.1	72.7	85	82.4	**83.5**	**82.6**	**91.7**
P3	47.9	28.1	24.5	63.8	32.2	10.5	9.7	55.1	**100**	9.7	11.8	54.9	86.2	**62.5**	**65.8**	**81.2**
P4	16	4.5	4.2	52.2	44.9	37.1	30.7	68.4	37.1	32.8	33.5	66.3	**68.4**	**63.8**	**63.1**	**81.8**
P5	47.1	18.2	16.6	58.8	61.4	51.9	42.8	74.9	66.8	24.4	27.8	62	**71.9**	**87.2**	**82.3**	**92.9**
P6	65.2	15.3	14.5	57.6	22.3	15.6	13.3	57.5	**97**	35.7	40.9	67.9	79.2	**70.2**	**68.1**	**84.8**
P7	41.3	7.5	7.2	53.7	44.1	5.5	5.3	52.7	12.5	4.8	8.1	52.3	**47.8**	**66.7**	**60.4**	**83.1**
P8	31.3	19.2	16.7	59.2	30.6	18.4	16	58.8	34.7	25	26.5	62.1	**38.6**	**61.1**	**49.6**	**79.2**
P9	41.7	12.2	11.4	56	2.9	1.6	1.4	50	5.4	3	3.3	51.1	**53.2**	**50.5**	**50.6**	**74.8**
P10	56.7	24.1	21.8	62	27.7	50.3	34.6	74.6	67.2	44.7	47.8	72.2	**74.9**	**68.1**	**68.6**	**84**
Avg.	46.1	18.4	16.5	59.1	34.1	29.8	22.9	64.5	53.7	31	32.2	65.3	**64.8**	**68.6**	**65.5**	**83.9**
Imp.	18.8	50.2	49	24.9	30.7	38.8	42.6	19.4	11.1	37.6	33.4	18.6	—	—	—	—
p-value	0.006**	0.000**	0.000**	0.000**	0.001**	0.000**	0.000**	0.000**	0.137	0.000**	0.000**	0.000**	—	—	—	—
Cohen's *d*	1.136 (L)	4.451 (L)	5.077 (L)	4.431 (L)	1.528 (L)	2.488 (L)	4.136 (L)	2.529 (L)	0.516 (M)	2.042 (L)	1.952 (L)	2.042 (L)	—	—	—	—

本文的 STRACE(AL+SSL) 模型有着显著的效果提升. 根据 Wilcoxon signed-rank (符号秩) 检验中差异的显著性 *p*-value 值和效应量 Cohen's *d* 值 (效应量小、中、大的区分临界点分别是: 0.20、0.50 和 0.80) 的结果, STRACE (AL+SSL) 相对于基于信息检索的矢量空间模型 IR(VSM) 和基于随机森林的机器学习模型 Baseline(RF) 有大幅度的提升. 在 10 个企业项目上, 与这两类传统模型相比, 综合指标 *F*2 分数分别平均提升了 49% 和 42.6%, AUC 值分别平均提升了 24.9% 和 19.4%. 同样的, 6 个开源项目数据上, STRACE(AL+SSL) 相比与这两种传统模型 *F*2 值提升了 44.5% 和 23.6%, AUC 值平均提升了 17.7% 和 13.7%. 这是因为, IR 模型是仅利用了软件制品间语义信息的推荐模型, 其对软件制品间过程信息的利用极其有限. 基于随机森林的 Baseline(RF) 模型受到本文第 2 节详细解析的数据稀疏性、数据不平衡等问题的影响, 且因为有标签样本中过少的实例数, 使得有监督模型很难学习到泛化能力较强的分类模型.

相对于半监督模型 SPLINT[44], 本文的 STRACE(AL+SSL) 模型在无论在开源项目 (+21.2%, +18.4% 和 +10.5%) 还是企业项目 (+37.6%、+33.4% 和+18.6%) 上, Recall、F2 及 AUC 值共 3 个指标上都有显著的提升. 这是因为 SPLINT 是只依赖于 Baseline(RF) 中构建的初始训练模型生成伪标签的半监督模型, 其生成的伪标签质量是难以保障的, 尤其在初始模型的训练效果不佳时, 选择伪标签样本加入训练集, 还可能使得模型整体效果不升反降. 值得注意的是在部分项目 (如开源项目 Drools, Groovy 等, 企业项目 P2, P3 及 P6 等) 上, STRACE(AL+SSL) 相比于 SPLINT, 其 *Precision* 存在不足, 其可能的原因之一是由于 STRACE(AL+SSL) 中同时结合了主动学习和半监督学习, 两个模块中都是用于补充更多无标签数据, 模型可以更好地捕获可追踪性样本空间的整体分布, 从而更加关注提升召回率 *Recall*, 相应地精确度 *Precision* 则会有所下降. 这恰好与恢复软件制品缺失链接, 预测结果得到较高的召回率[40]的期望相吻合. 故 STRACE(AL+SSL) 更加符合本文的任务场景.

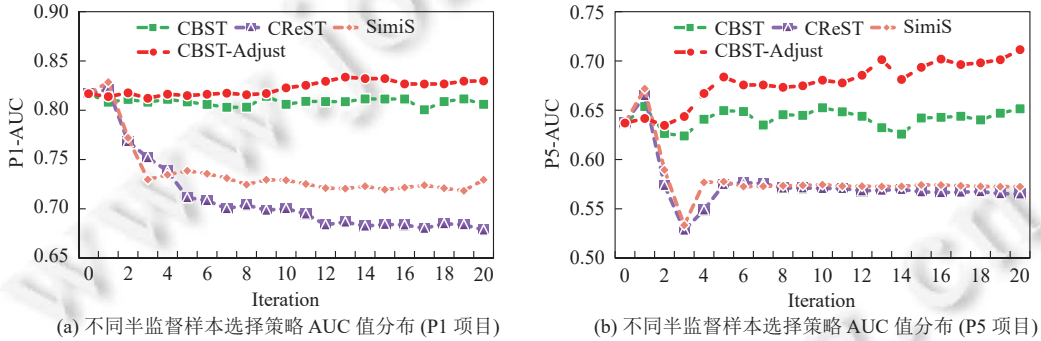
在本文的 STRACE(AL+SSL) 框架中, 为方便分析不同改进策略带来的影响, 基础分类模型算法仅选择了与 SPLINT 和 Baseline(RF) 一致的随机森林算法, 所汇报的也是在同一参数配置下, 综合多次实验结果取得最优平均值的结果. 在其他特定任务场景中, 也可替换任意改进策略的算法, 包括将随机森林模型替换为深度学习模型, 充分学习大规模数据集信息.

总的来说, 基于 6 个开源项目和 10 个企业项目的实验结果表明, STRACE(AL+SSL) 显著优于传统的 IR 方法和随机森林 Baseline(RF) 方法. 同时, 对于半监督学习强依赖于伪标签质量的问题, STRACE(AL+SSL) 框架具有针对性的提升效果.

RQ2: STRACE(AL+SSL) 框架中各阶段 (半监督及主动学习) 最佳的样本选择策略是哪种?

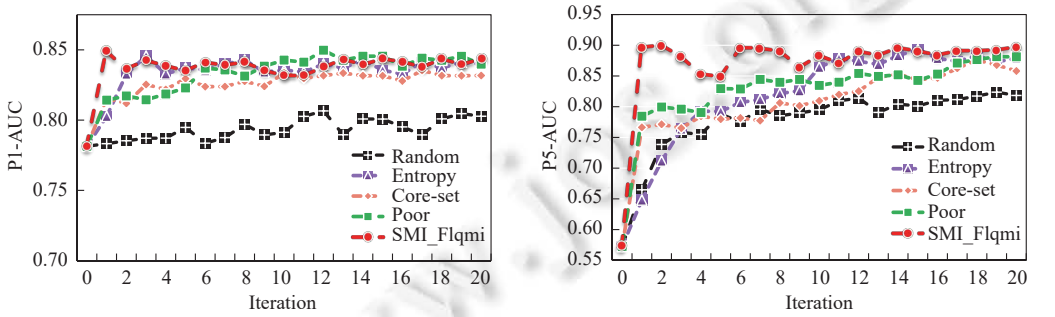
为了回答该问题, 本文在 STRACE(AL+SSL) 框架中半监督及主动学习阶段选择无标签样本时, 对比了不同策略的效果. 半监督学习策略中对比了平衡采样策略 CBST[51,52]、CReST[53]、SimiS[54]与本文改进后的 CBST-Adjust.

主动学习策略中对比了传统的基于熵 (Entropy)[57,58,60]的不确定性策略 (选择分类器决策边界附近最具不确定性的链接样本), 基于特征空间覆盖的 Core-set 策略[69](选择尽可能覆盖样本特征空间的待标注链接), 偏向少类样本的 Poor 策略[70](选择样本空间中相对于更靠近少类“真”链接的待标注链接) 和本文参考的基于子模块互信息的 SMI_Flqmi 策略[63,64,67](选择既偏向于少类“真”链接样本、又尽可能覆盖样本特征空间的待标注链接) 等. 这些策略在文本分类及图像视觉领域的工作中都有很好的预测效果. 如图 12 及图 13 所示, 本文选择了 10 个企业项目中制品规模最大 (总样本量最大) 的应用程序类项目 P1 和规模较小的服务平台类项目 P5 作为例子, 回答本文第 2 个研究问题.



(a) 不同半监督样本选择策略 AUC 值分布 (P1 项目) (b) 不同半监督样本选择策略 AUC 值分布 (P5 项目)

图 12 半监督学习样本选择策略对比



(a) 不同主动学习样本选择策略 AUC 值分布 (P1 项目) (b) 不同主动学习样本选择策略 AUC 值分布 (P5 项目)

图 13 主动学习样本选择策略对比

如图 12 所示, 可以看出在无论在较大规模的项目 P1 还是较小规模的项目 P5 上, 在 STRACE(AL+SSL) 框架下, 使用本文提出的改进后的 CBST-Adjust 作为半监督样本选择策略可以得到较好的结果. 与 CBST 相比, CBST-Adjust 由于进一步限制了多类伪标签样本, 减轻了所选择伪标签中的类别不平衡问题的严重程度. 此外, 值得注意

的是, 虽然 CReST 和 SimiS 半监督采样策略在首轮迭代对模型的改进明显优于 CBST 和 CBST-Adjust 策略 (学习到了更多的少类样本), 但随着迭代轮数的增加, 前两种策略只关注样本平衡性, 倾向于选择更多的少类伪标签样本, 但忽略了伪标签样本中少类样本“数量-质量”的 trade-off, 因此在后续迭代中模型效果反而下降得愈加明显. 而本文改进的 CBST-Adjust 中既考虑了较高质量的伪标签, 又引入了伪标签的权重机制, 从而缓解了所选择伪标签随数量增加其低质量对模型的负面影响. 故在 P1 和 P5 两个项目中, 使用 CBST-Adjust 策略后, 可追踪性模型 AUC 值随着迭代轮数的增加, 有稳步提升的趋势.

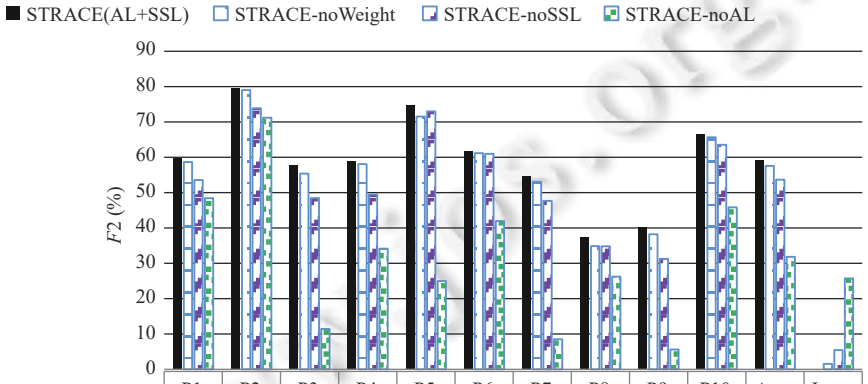
如图 13 所示, 在 STRACE(AL+SSL) 框架中的主动学习阶段, 相对于随机 (Random) 采样策略, 无论是主流的基于熵 (Entropy) 不确定性策略, 还是基于特征空间覆盖的 Core-set 策略或偏向少类的 Poor 策略, 抑或是基于子模块互信息的 SMI_Flqmi 策略都有明显的优势. 值得注意的是, 在各种半监督学习的采样策略中, 本文希望得到随着迭代进行模型指标稳步提升的效果. 与之不同的是, 主动学习阶段涉及标注成本的问题, 并且模型性能并不随标注成本单调递增, 这种趋势在其他开源项目的追踪性恢复任务上[57]及其他场景的任务[71]也是类似的. 故本文在主动学习阶段更加希望选择投入较少成本 (即标注较少样本) 即可得到较好模型效果的样本选择策略.

在本文的可追踪性恢复任务中, 有标签样本少且类别不平衡问题也较为严重, 因此本文选择既相似于少类的“真”链接样本、又尽可能覆盖样本空间的待标注链接样本的 SMI_Flqmi 策略在初始迭代, 也就是仅标注 1%–2% 左右的标签后就可以得到相对于其他主动学习样本选择策略更优的模型效果. 而其他策略往往需要迭代更多轮, 意味着需要投入更多成本, 标注更多样本.

因此, 在本文的软件可追踪性分类任务中, 相对于大多数主动学习样本选择策略, STRACE(AL+SSL) 框架使用 SMI_Flqmi 样本选择策略有较大优势.

RQ3: STRACE(AL+SSL) 框架中各模块 (半监督、主动学习、权重机制) 的贡献是怎样的?

为了解 STRACE(AL+SSL) 框架中各模块的贡献, 本文进一步补充了消融实验, 实验结果如图 14 所示, 展示了 10 个企业项目进行 10 次重复实验, 每次实验训练迭代 20 次的平均值. 柱状图中第 1 列为 STRACE(AL+SSL) 完整框架的实验效果, 第 2 列为删掉伪标签权重机制后的结果 STRACE-noWeight, 在 10 个企业项目上, 权重机制都贡献了平均约 1.54% 的 $F2$ 值的提升. STRACE-noSSL 为删掉半监督学习模块后的结果, 在 10 个企业项目上, 半监督学习模块平均贡献了约 5.44% 的 $F2$ 值的提升. 进一步地对比 STRACE-noAL 的结果, 不难发现, 主动学习的引入为 STRACE(AL+SSL) 可追踪性恢复模型的性能的提升贡献最为显著, $F2$ 值平均提升约 25.73%. 这表明使得模型学习到具有代表性的, 值得标注的样本对模型的性能提升更为有效.



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Avg.	Imp.
STRACE(AL+SSL)	59.61	79.54	57.68	58.88	74.68	61.84	54.74	37.26	40.33	66.48	59.10	0.00
STRACE-noWeight	58.65	79.02	55.38	58.07	71.52	61.17	53.05	34.87	38.22	65.66	57.56	1.54
STRACE-noSSL	53.57	73.88	48.46	49.49	73.00	60.97	47.65	34.81	31.26	63.54	53.66	5.44
STRACE-noAL	48.42	71.19	11.43	34.16	24.98	41.95	8.51	26.23	5.61	45.83	31.83	25.73

图 14 消融实验结果

5 讨 论

5.1 挑战和约束

本文采用主动学习和半监督学习结合式的思路构建了 STRACE(AL+SSL) 框架. 这两种技术类属于弱监督学习, 在软件可追踪性恢复任务的实践场景中仍然面临一些挑战和约束.

首先一个挑战是标注成本, 尽管主动学习旨在最大程度地减少标注成本, 但在实际应用中, 获取标注仍然需要一定的资源和成本. 因此, 在选择哪些样本进行主动学习时, 需要在成本和性能之间进行权衡. 本文引入主动学习策略时, 既要考虑到可追踪性任务面临的数据稀疏性和不平衡性等各项样本集中已存在的数据挑战对模型带来的影响, 又要确保在实践中标注的成本是尽可能小, 因此本文在 STRACE(AL+SSL) 方法中参考了 Kothawade 等人[63,64]提出的子模块互信息 (SMI) 函数[65,66]. 本文 RQ2 中的实验结果也表明了 STRACE(AL+SSL) 方法采用基于子模块互信息 (SMI) 函数的主动学习策略有助于缩短迭代次数 (即标注成本) 的同时, 也能得到较好的模型效果. 其次是对标注人员的要求也是 STRACE(AL+SSL) 方法实践中的难点, 不可否认的是区别于通用图像分类任务对对标注人员的领域知识没有太高要求. 而软件可追踪性恢复任务中, 制品关联链接的标注往往需要是第一手负责开发者, 测试人员等该制品的数据生产者来标注样本, 这无疑增加了标注成本和复杂性.

初始训练集质量不可避免会影响 STRACE(AL+SSL) 方法的效果, 主动学习及半监督学习选择的样本能否对于提升模型准确性有效依赖于基于初始训练集得到的初始模型的准确性. 举例而言, 如图 15 所示, 通过对比不同类不平衡性的初始训练集 (P1 项目为例) 条件下, STRACE(AL+SSL) 模型 $F2$ 值差异, 可以观察到, 当初始训练集类不平衡变得越高时, STRACE (AL+SSL) 方法得到的初始模型效果越差. 即使后续训练迭代多次, 类不平衡性最高的训练集得到的模型效果依然不佳, 即折线 2×ImR.始终在最下面. 当初始训练集类不平衡性有所下降时, 如折线 1/4×ImR.效果最佳, ImR.表示 P1 项目初始训练集的不平衡性. 这表明了初始训练集的类不平衡性会严重影响 STRACE(AL+SSL) 的效果. 本文选择了 SMOTE 算法对训练集 train_set 重采样来缓解初始训练集样本类不平衡问题, 在后续迭代中, STRACE(AL+SSL) 的半监督模块采用动态调整少类无标签样本的阈值, 主动学习模块则采用子模块互信息函数来选择样本集中与“真” 链接样本模块相似的样本. 除了以上方法之外业界也有很多其他方法可供参, 如代价敏感学习, 类权重调整[72]等.

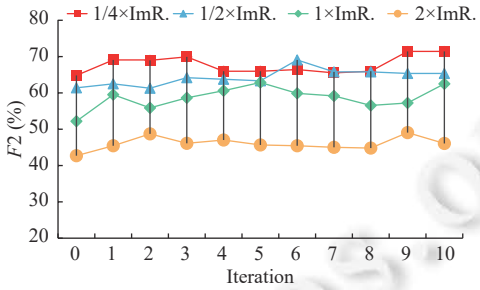


图 15 初始训练集类不平衡性影响

虽然在本文的软件可追踪性的恢复任务场景中, 主动学习结合半监督学习的框架有着显著的效果提升, 但在安全性要求较高的任务场景下, 如自动驾驶, 医学诊断等领域, 伪标签的生成及使用还需要慎重考虑.

5.2 效度威胁

针对内部效度 (internal validity) 而言, 本文实验数据的准备可能带来一定程度的有效性威胁. 并非所有 commit 都与缺陷 issue 存在潜在链接关系, 本文在参考 Rath 等人[10]提出的软件制品跟踪对匹配函数 *is_candidate* 的基础上, 考虑到企业项目的业务属性、团队大小及项目规模等差异, 动态调整了时间参数 ε. 此外本文观察到企业项目 commit 一般只与一个制品 ID 关联, 因此若某 commit 已有明确关联制品 ID, 本文会删除其他缺陷与此 commit 形成的候选“假”链接, 从而降低样本不平衡性. 为了训练和验证可追踪性模型, 需要构建制品间存在“真”或

“假”链接的 ground-truth 数据集, 本文抽取了缺陷管理系统中 10 个企业项目在规定时间内发布的完整版本的缺陷报告记录, 选择了明确“已修复 (resolved)”且“已关闭 (closed)”的缺陷, 从代码仓中选择成功合入 (merged) 的 commit 记录, 并以开发人员在 commit 属性表中关联的制品 ID 记录作为确定样本 ground-truth 标签的依据. 样本 ground-truth 标注过程中的毋庸置疑会存在标签准确性的问题. 本文为了模拟真实数据集上有标签样本和无标签样本的真实分布, 将 100% 的有标签集合 (ground-truth) 视为每个项目的全部数据集, 并从中随机选择 30% 的样视为初始有标签样本, 同时也是初始训练集, 剩余 70% 视作无标签样本, 来支持本文 STRACE(AL+SSL) 方法的循环迭代训练. 其中无论是初始的 30% 的有标签训练集, 还是 70% 的无标签数据集, 其标签的标注结果都可能由于开发人员人工打错 tag 等原因引入的标签错误 (label error) 问题则会对模型效果产生影响. 如何保障标签结果准确性的工作还值得进一步研究与探索. 本文建议可参考人工和自动化双保障的手段, 如采用增加人力成本来保障标注结果准确性, 多人标注交叉验证, 多次标注降低误差, 领域专家审查[73,74] 等. 当然还可以尝试自动化的手段, 如一致性检查, 标注模型融合等[75,76] 来识别和修正标签错误 (label error) 问题.

为了应对结论效度 (conclusion validity) 威胁, 本文首先基于合作企业的所有可访问项目 (50 余个) 的历史数据进行了前期调研工作, 并结合内部不同角色工程师 (业务项目工程师、开发人员、测试人员和软件资源库平台工程师) 的反馈, 汇总了软件可追踪性现状及潜在的关键挑战, 以确保调研工作的系统性和全面性. 在后续实验验证环节结合了假设检验方法和 Cohen’s d 效应量来评估本文 STRACE 模型与之前方法之间的显著性差异, 从而能够保证结论的正确性.

为了缓解外部效度 (external validity), 本文的实验样本, 即 10 个企业项目仅来自一家合作企业, 软件制品选择了 issue-commit 两种类别, 这可能对本文实验的外部有效性带来威胁. 对此, 本文在实验的数据准备阶段从合作企业的 50 余个可访问项目中, 选择了来自不同业务属性且有连续发布正式版本的 10 个项目 (项目的总样本范围在 1 万至近百万个 issue-commit 跟踪对实例), 以确保实验样本的多样性和代表性. 尽管本文仅报告了所提出的 STRACE (AL+SSL) 框架应用于某软件企业的商业项目的 issue-commit 间的“真”或“假”链接分类任务, 但该框架有潜力被应用于其他多种软件制品间的可追踪性恢复任务. 其他软件组织在使用本文方法时, 可以参考本文分析的几点关键挑战的经验, 根据自身数据情况和特性, 灵活地调整框架中的各个组件, 如分类基础模型及采样策略, 以更好地实践本文所提供的可追踪性恢复框架.

6 总 结

自动化的软件可追踪性恢复方法旨在解决软件过程制品之间潜在追踪链接缺失的问题. 通过对软件可追踪性质量现状进行调研, 本文发现企业项目中制品间可追踪性低质量问题更为严峻, 例如软件制品之间存在缺乏关联、延迟关联、错误关联和不完整关联等问题. 因此, 对可追踪性进行自动化恢复的需求变得更加紧迫. 为了解工业界场景中软件过程制品间可追踪性的现状及可追踪性模型表现不佳的原因, 本文前期展开了现状调研与实验案例探索研究, 进而分析识别出了原始数据低质量、数据稀疏及样数据不平衡等 3 点关键挑战. 本文所提出的 STRACE (AL+SSL) 框架结合了主动学习和半监督学习两种学习策略, 致力于解决可追踪性模型表现不佳的问题. 针对这 3 点挑战, 首先, 在模型训练阶段引入半监督学习策略, 利用无标签样本进行学习, 解决样本规模稀疏性问题. 同时, 在半监督样本选择机制中, 提出了调整后的类平衡自训练 (CBST-Adjust) 样本选择策略, 重点解决数据不平衡问题. 通过引入伪标签权重机制, 在选择伪标签时考虑初始训练模型精度和伪标签样本可靠性的影响, 为伪标签样本赋予不同的权重, 解决半监督模型因引入低质量伪标签而导致的效果不稳定和伪标签利用率不高的问题. 其次, 在模型训练阶段引入基于主动学习子模块互信息的 SMI_Flqmi 策略, 对初始训练集进行重塑, 并与半监督学习策略相结合, 重点减轻初始训练集低质量的影响, 提高半监督学习生成的伪标签质量. 本文基于 6 个开源项目及 10 个企业项目展开了多组对比实验, 实验结果表明, STRACE(AL+SSL) 显著优于传统的 IR 方法和随机森林 Baseline (RF) 方法 ($F2$ 值平均提升范围为 20%–40%). 对于半监督学习模型 SPLINT 强依赖于伪标签质量的问题, STRACE (AL+SSL) 框架具有针对性地提升效果 ($F2$ 值平均提升 18%–30%). 从而验证了所提框架在真实企业项目软件可

追踪性恢复任务上的有效性.

未来工作一方面将持续验证本文提出的 STRACE(AL+SSL) 框架在其他软件企业和开源社区项目上的有效性. 同时将持续拓展和优化框架, 适配多种制品间的可追踪性恢复任务. 另一方面, 未来工作将持续深入研究链接标注质量及成本问题, 提供更符合实践需求的软件可追踪性质量优化方案.

References:

[1] Watkins R, Neal M. Why and how of requirements tracing. IEEE Software, 1994, 11(4): 104–106. [doi: 10.1109/52.300100]

[2] Kukkanen J, Väkeväinen K, Kauppinen M, Uusitalo E. Applying a systematic approach to link requirements and testing: A case study. In: Proc. of the 16th Asia-Pacific Software Engineering Conf. Batu Ferringhi: IEEE, 2009. 482–488. [doi: 10.1109/apsec.2009.62]

[3] De Toledo SS, Martini A, Sjøberg DIK. Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study. Journal of Systems and Software, 2021, 177: 110968. [doi: 10.1016/j.jss.2021.110968]

[4] Dasanayake S, Aaramaa S, Markkula J, Oivo M. Impact of requirements volatility on software architecture: How do software teams keep up with ever-changing requirements? Journal of Software: Evolution and Process, 2019, 31(6): e2160. [doi: 10.1002/smr.2160]

[5] Fucci D, Alégroth E, Axelsson T. When traceability goes awry: An industrial experience report. Journal of Systems and Software, 2022, 192: 111389. [doi: 10.1016/j.jss.2022.111389]

[6] Cleland-Huang J, Chang CK, Christensen M. Event-based traceability for managing evolutionary change. IEEE Trans. on Software Engineering, 2003, 29(9): 796–810. [doi: 10.1109/tse.2003.1232285]

[7] Hayes JH, Dekhtyar A, Sundaram SK, Holbrook EA, Vadlamudi S, April A. Requirements tracing on target (retro): Improving software maintenance through traceability recovery. Innovations in Systems and Software Engineering, 2007, 3(3): 193–202. [doi: 10.1007/s11334-007-0024-1]

[8] Mills C, Escobar-Avila J, Haiduc S. Automatic traceability maintenance via machine learning classification. In: Proc. of the 2018 IEEE Int'l Conf. on Software Maintenance and Evolution. Madrid: IEEE, 2018. 369–380. [doi: 10.1109/icsme.2018.00045]

[9] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 2002, 16(1): 321–357. [doi: 10.1613/jair.953]

[10] Rath M, Rendall J, Guo JLC, Cleland-Huang J, Mäder P. Traceability in the wild: Automatically augmenting incomplete trace links. In: Proc. of the 40th Int'l Conf. on Software Engineering. Gothenburg: ACM, 2018. 834–845. [doi: 10.1145/3180155.3180207]

[11] Kaushik N, Tahvildari L, Moore M. Reconstructing traceability between bugs and test cases: An experimental study. In: Proc. of the 18th Working Conf. on Reverse Engineering. Limerick: IEEE, 2011. 411–414. [doi: 10.1109/wcre.2011.58]

[12] Moran K, Palacio DN, Bernal-Cárdenas C, McCrystal D, Poshyvanyk D, Shenefiel C, Johnson J. Improving the effectiveness of traceability link recovery using hierarchical Bayesian networks. In: Proc. of the 42nd Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 873–885. [doi: 10.1145/3377811.3380418]

[13] Guo J, Cheng JH, Cleland-Huang J. Semantically enhanced software traceability using deep learning techniques. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Software Engineering. Buenos Aires: IEEE, 2017. 3–14. [doi: 10.1109/icse.2017.9]

[14] Rodriguez AD, Cleland-Huang J, Falessi D. Leveraging intermediate artifacts to improve automated trace link retrieval. In: Proc. of the 2021 IEEE Int'l Conf. on Software Maintenance and Evolution. Luxembourg: IEEE, 2021. 81–92. [doi: 10.1109/icsme52107.2021.00014]

[15] Gotel OCZ, Finkelstein CW. An analysis of the requirements traceability problem. In: Proc. of the 1994 IEEE Int'l Conf. on Requirements Engineering. Colorado Springs: IEEE, 1994. 94–101. [doi: 10.1109/icre.1994.292398]

[16] Gotel O, Cleland-Huang J, Hayes JH, Zisman A, Egyed A, Grünbacher P, Dekhtyar A, Antoniol G, Maletic J. The grand challenge of traceability (v1.0). In: Cleland-Huang J, Gotel O, Zisman A, eds. Software and Systems Traceability. London: Springer, 2012. 343–409. [doi: 10.1007/978-1-4471-2239-5_16]

[17] Rempel P, Mäder P. Preventing defects: The impact of requirements traceability completeness on software quality. IEEE Trans. on Software Engineering, 2017, 43(8): 777–797. [doi: 10.1109/tse.2016.2622264]

[18] Cleland-Huang J. Traceability in agile projects. In: Cleland-Huang J, Gotel O, Zisman A, eds. Software and Systems Traceability. London: Springer, 2012. 265–275. [doi: 10.1007/978-1-4471-2239-5_12]

[19] Neumuller C, Grunbacher P. Automating software traceability in very small companies: A case study and lessons learned. In: Proc. of the 21st IEEE/ACM Int'l Conf. on Automated Software Engineering. Tokyo: IEEE, 2006. 145–156. [doi: 10.1109/ase.2006.25]

[20] Panis MC. Successful deployment of requirements traceability in a commercial engineering organization … really. In: Proc. of the 18th IEEE Int'l Requirements Engineering Conf. Sydney: IEEE, 2010. 303–307. [doi: 10.1109/re.2010.43]

[21] Rath M, Lo D, Mäder P. Analyzing requirements and traceability information to improve bug localization. In: Proc. of the 15th Int'l Conf. on Mining Software Repositories. Gothenburg: ACM, 2018. 442–453. [doi: 10.1145/3196398.3196415]

[22] Parizi RM, Lee SP, Dabbagh M. Achievements and challenges in state-of-the-art software traceability between test and code artifacts. IEEE Trans. on Reliability, 2014, 63(4): 913–926. [doi: 10.1109/tr.2014.2338254]

[23] Ali NB, Petersen K. A consolidated process for software process simulation: State of the art and industry experience. In: Proc. of the 38th Euromicro Conf. on Software Engineering and Advanced Applications. Cesme: IEEE, 2012. 327–336. [doi: 10.1109/seaa.2012.69]

[24] Li Y, Zhang H, Dong LM, Liu BH, Ma JY. Constructing a hybrid software process simulation model in practice: An exemplar from industry. In: Proc. of the 2020 Int'l Conf. on Software and System Processes. Seoul: ACM, 2020. 135–144. [doi: 10.1145/3379177.3388906]

[25] Chen XF, Grundy J. Improving automated documentation to code traceability by combining retrieval techniques. In: Proc. of the 26th IEEE/ACM Int'l Conf. on Automated Software Engineering. Lawrence: IEEE, 2011. 223–232. [doi: 10.1109/ase.2011.6100057]

[26] Winkler S, Von Pilgrim J. A survey of traceability in requirements engineering and model-driven development. Software & Systems Modeling, 2010, 9(4): 529–565. [doi: 10.1007/s10270-009-0145-0]

[27] Rath M, Mäder P. The SEOSS 33 dataset——Requirements, bug reports, code history, and trace links for entire projects. Data in Brief, 2019, 25: 104005. [doi: 10.1016/j.dib.2019.104005]

[28] Egyed A, Graf F, Grünbacher P. Effort and quality of recovering requirements-to-code traces: Two exploratory experiments. In: Proc. of the 18th IEEE Int'l Requirements Engineering Conf. Sydney: IEEE, 2010. 221–230. [doi: 10.1109/re.2010.34]

[29] Borg M, Runeson P, Ardö A. Recovering from a decade: A systematic mapping of information retrieval approaches to software traceability. Empirical Software Engineering, 2014, 19(6): 1565–1616. [doi: 10.1007/s10664-013-9255-y]

[30] Corallo A, Latino ME, Menegoli M, Pontrandolfo P. A systematic literature review to explore traceability and lifecycle relationship. Int'l Journal of Production Research, 2020, 58(15): 4789–4807. [doi: 10.1080/00207543.2020.1771455]

[31] Zogaan W, Sharma P, Mirahkorli M, Arnaoudova V. Datasets from fifteen years of automated requirements traceability research: Current state, characteristics, and quality. In: Proc. of the 25th IEEE Int'l Requirements Engineering Conf. Lisbon: IEEE, 2017. 110–121. [doi: 10.1109/re.2017.80]

[32] Aung TWW, Huo H, Sui YL. A literature review of automatic traceability links recovery for software change impact analysis. In: Proc. of the 28th Int'l Conf. on Program Comprehension. Seoul: ACM, 2020. 14–24. [doi: 10.1145/3387904.3389251]

[33] Antoniol G, Canfora G, De Lucia A, Merlo E. Recovering code to documentation links in OO systems. In: Proc. of the 6th Working Conf. on Reverse Engineering. Atlanta: IEEE, 1999. 136–144. [doi: 10.1109/wcre.1999.806954]

[34] Zhai YP, Hong M, Yang QH. Research on traceability of functional requirements to test case. Computer Science, 2017, 44(11A): 480–484 (in Chinese with English abstract).

[35] Antoniol G, Canfora G, Casazza G, De Lucia A, Merlo E. Recovering traceability links between code and documentation. IEEE Trans. on Software Engineering, 2002, 28(10): 970–983. [doi: 10.1109/tse.2002.1041053]

[36] Marcus A, Maletic JI, Sergeyev A. Recovery of traceability links between software documentation and source code. Int'l Journal of Software Engineering and Knowledge Engineering, 2005, 15(5): 811–836. [doi: 10.1142/S0218194005002543]

[37] Asuncion HU, Asuncion AU, Taylor RN. Software traceability with topic modeling. In: Proc. of the 32nd ACM/IEEE Int'l Conf. on Software Engineering. Cape Town: ACM, 2010. 95–104. [doi: 10.1145/1806799.1806817]

[38] Abadi A, Nisenson M, Simionovici Y. A traceability technique for specifications. In: Proc. of the 16th IEEE Int'l Conf. on Program Comprehension. Amsterdam: IEEE, 2008. 103–112. [doi: 10.1109/icpc.2008.30]

[39] Cleland-Huang J, Czauderna A, Gibiec M, Emenecker J. A machine learning approach for tracing regulatory codes to product specific requirements. In: Proc. of the 32nd ACM/IEEE Int'l Conf. on Software Engineering. Cape Town: ACM, 2010. 155–164. [doi: 10.1145/1806799.1806825]

[40] Lin JF, Liu YL, Zeng QK, Jiang M, Cleland-Huang J. Traceability transformed: Generating more accurate links with pre-trained BERT models. In: Proc. of the 43rd IEEE/ACM Int'l Conf. on Software Engineering. Madrid: IEEE, 2021. 324–335. [doi: 10.1109/icse43902.2021.00040]

[41] Ruan H, Chen BH, Peng X, Zhao WY. DEEPLINK: Recovering issue-commit links based on deep learning. Journal of Systems and Software, 2019, 158: 110406. [doi: 10.1016/J.JSS.2019.110406]

[42] Hammoudi M, Mayr-Dorn C, Mashkoor A, Egyed A. A traceability dataset for open source systems. In: Proc. of the 18th IEEE/ACM Int'l Conf. on Mining Software Repositories. Madrid: IEEE, 2021. 555–559. [doi: 10.1109/msr52588.2021.00073]

[43] Maro S, Staron M, Steghöfer JP. Challenges of establishing traceability in the automotive domain. In: Proc. of the 9th Int'l Conf. on Software Quality. Vienna: Springer, 2017. 153–172. [doi: 10.1007/978-3-319-49421-0_11]

[44] Dong LM, Zhang H, Liu W, Weng ZL, Kuang HY. Semi-supervised pre-processing for learning-based traceability framework on real-world software projects. In: Proc. of the 30th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Singapore: ACM, 2022. 570–582. [doi: 10.1145/3540250.3549151]

[45] Le TDB, Linares-Vasquez M, Lo D, Poshyvanyk D. RCLinker: Automated linking of issue reports and commits leveraging rich contextual information. In: Proc. of the 23rd IEEE Int'l Conf. on Program Comprehension. Florence: IEEE, 2015. 36–47. [doi: 10.1109/icpc.2015.13]

[46] Cavnar WB. Using an n-gram-based document representation with a vector processing retrieval model. In: Harman DK, ed. Proc. of the 3rd Text Retrieval Conf. (TREC-3). Gaithersburg: National Institute of Standards and Technology, 1994. 269–278.

[47] Gethers M, Oliveto R, Poshyvanyk D, De Lucia A. On integrating orthogonal information retrieval methods to improve traceability recovery. In: Proc. of the 27th IEEE Int'l Conf. on Software Maintenance. Williamsburg: IEEE, 2011. 133–142. [doi: 10.1109/icsm.2011.6080780]

[48] Chen BH, Chen LL, Zhang C, Peng X. BuildFast: History-aware build outcome prediction for fast feedback and reduced cost in continuous integration. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering. Melbourne: ACM, 2020. 42–53. [doi: 10.1145/3324884.3416616]

[49] Sun Y, Wang Q, Yang Y. FRLink: Improving the recovery of missing issue-commit links by revisiting file relevance. Information and Software Technology, 2017, 84: 33–47. [doi: 10.1016/j.infsof.2016.11.010]

[50] Sohn K, Berthelot D, Li CL, Zhang ZZ, Carlini N, Cubuk ED, Kurakin A, Zhang H, Raffel C. FixMatch: Simplifying semi-supervised learning with consistency and confidence. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 596–608.

[51] Zou Y, Yu ZD, Kumar BVK, Wang JS. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: Proc. of the 15th European Conf. on Computer Vision. Munich: Springer, 2018. 297–313. [doi: 10.1007/978-3-030-01219-9_18]

[52] Zou Y, Yu ZD, Liu XF, Kumar BVKV, Wang JS. Confidence regularized self-training. In: Proc. of the 2019 IEEE/CVF Int'l Conf. on Computer Vision. Seoul: IEEE, 2019. 5981–5990. [doi: 10.1109/iccv.2019.00608]

[53] Wei C, Sohn K, Mellina C, Yuille A, Yang F. CReST: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In: Proc. of the 2021 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Nashville: IEEE, 2021. 10852–10861. [doi: 10.1109/cvpr46437.2021.01071]

[54] Chen H, Fan Y, Wang YD, Wang JD, Schiele B, Xie X, Savvides M, Raj B. An embarrassingly simple baseline for imbalanced semi-supervised learning. arXiv:2211.11086, 2022.

[55] Xu Y, Shang L, Ye JX, Qian Q, Li YF, Sun BG, Li H, Jin R. Dash: Semi-supervised learning with dynamic thresholding. In: Proc. of the 38th Int'l Conf. on Machine Learning. ICML, 2021. 11525–11536.

[56] Zhang BW, Wang YD, Hou WX, Wu H, Wang JD, Okumura M, Shinozaki T. FlexMatch: Boosting semi-supervised learning with curriculum pseudo labeling. In: Proc. of the 34th Annual Conf. on Neural Information Processing Systems. 2021. 18408–18419.

[57] Mills C, Escobar-Avila J, Bhattacharya A, Kondyukov G, Chakraborty S, Haiduc S. Tracing with less data: Active learning for classification-based traceability link recovery. In: Proc. of the 2019 IEEE Int'l Conf. on Software Maintenance and Evolution. Cleveland: IEEE, 2019. 103–113. [doi: 10.1109/icsme.2019.00020]

[58] Du TB, Shen GH, Huang ZQ, Yu YS, Wu DX. Automatic traceability link recovery via active learning. Frontiers of Information Technology & Electronic Engineering, 2020, 21(8): 1217–1225. [doi: 10.1631/fitee.1900222]

[59] Tharwat A, Schenck W. A survey on active learning: State-of-the-art, practical challenges and research directions. Mathematics, 2023, 11(4): 820. [doi: 10.3390/math11040820]

[60] Prenner JA, Robbes R. Making the most of small software engineering datasets with modern machine learning. IEEE Trans. on Software Engineering, 2022, 48(12): 5050–5067. [doi: 10.1109/tse.2021.3135465]

[61] Lewis DD, Catlett J. Heterogeneous uncertainty sampling for supervised learning. In: Cohen WW, Hirsh H, eds. Machine Learning: Proc. of the 11th Int'l Conf. New Brunswick: Elsevier, 1994. 148–156. [doi: 10.1016/b978-1-55860-335-6.50026-x]

[62] Scheffer T, Decomain C, Wrobel S. Active hidden Markov models for information extraction. In: Proc. of the 4th Int'l Symp. on Intelligent Data Analysis. Cascais: Springer, 2001. 309–318. [doi: 10.1007/3-540-44816-0_31]

[63] Kothawade S, Reddy PK, Ramakrishnan G, Iyer R. BASIL: Balanced active semi-supervised learning for class imbalanced datasets. arXiv:2203.05651, 2022.

[64] Kothawade S, Ghosh S, Shekhar S, Xiang Y, Iyer R. Talisman: Targeted active learning for object detection with rare classes and slices using submodular mutual information. In: Proc. of the 17th European Conf. on Computer Vision. Tel Aviv: Springer, 2022. 1–16. [doi: 10.1007/978-3-031-19839-7_1]

[65] Gupta A, Levin R. The online submodular cover problem. In: Proc. of the 2020 ACM-SIAM Symp. on Discrete Algorithms. Salt Lake City: SIAM, 2020. 1525–1537. [doi: 10.1137/1.9781611975994.94]

[66] Iyer RK, Khargoankar N, Bilmes JA, Asanani H. Submodular combinatorial information measures with applications in machine learning. In: Proc. of the 32nd Algorithmic Learning Theory. ALT, 2021. 722–754.

[67] Kothawade S, Savarkar A, Iyer V, Ramakrishnan G, Iyer R. CLINICAL: Targeted active learning for imbalanced medical image classification. In: Proc. of the 1st Workshop on Medical Image Learning with Limited and Noisy Data. Singapore: Springer, 2022. 119–129. [doi: 10.1007/978-3-031-16760-7_12]

[68] Chen H, Tao R, Fan Y, Wang YD, Wang JD, Schiele B, Xie X, Raj B, Savvides M. SoftMatch: Addressing the quantity-quality trade-off in semi-supervised learning. arXiv:2301.10921, 2023.

[69] Sener O, Savarese S. Active learning for convolutional neural networks: A core-set approach. In: Proc. of the 6th Int'l Conf. on Learning Representations. Vancouver: OpenReview.net, 2018.

[70] Belouadah E, Popescu A, Aggarwal U, Saci L. Active class incremental learning for imbalanced datasets. In: Proc. of the 2020 European Conf. on Computer Vision. Glasgow: Springer, 2020. 146–162. [doi: 10.1007/978-3-030-65414-6_12]

[71] Kaplan J, McCandlish S, Henighan T, Brown TB, Chess B, Child R, Gray S, Radford A, Wu J, Amodei D. Scaling laws for neural language models. arXiv:2001.08361, 2020.

[72] Cao KD, Wei C, Gaidon A, Arechiga N, Ma TY. Learning imbalanced datasets with label-distribution-aware margin loss. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 140.

[73] Goh HW, Mueller J. ActiveLab: Active learning with re-labeling by multiple annotators. arXiv:2301.11856, 2023.

[74] Goh HW, Tkachenko U, Mueller J. CROWDLAB: Supervised learning to infer consensus labels and quality scores for data with multiple annotators. arXiv:2210.06812, 2022.

[75] Northcutt C, Jiang L, Chuang I. Confident learning: Estimating uncertainty in dataset labels. Journal of Artificial Intelligence Research, 2021, 70: 1373–1411. [doi: 10.1613/jair.1.12125]

[76] Chong D, Hong J, Manning C. Detecting label errors by using pre-trained language models. In: Proc. of the 2022 Conf. on Empirical Methods in Natural Language Processing. Abu Dhabi: ACL, 2022. 9074–9091. [doi: 10.18653/v1/2022.emnlp-main.618]

附中文参考文献:

[34] 翟宇鹏, 洪玫, 杨秋辉. 功能需求到测试用例的可追溯性研究. 计算机科学, 2017, 44(11A): 480–484.

董黎明(1994－), 女, 博士, 主要研究领域为软件工程, 软件研发效能, 软件过程, 软件可追踪性, 可信人工智能.

孟庆龙(1999－), 男, 硕士生, 主要研究领域为软件工程, 软件研发效能.

张贺(1971－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为软件工程, 开发运维一体化, 软件研发效能, 软件安全, 经验及循证软件工程, 区块链.

匡宏宇(1985－), 男, 博士, 助理研究员, CCF 专业会员, 主要研究领域为软件可追踪性, 自动化软件可追踪分析, 情绪分析, 文本分析, 程序分析.