软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

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

代码克隆相关研究综述*

张迅晖[1], 王 涛[1], 余 跃[1], 钟 岩[1], 张晏芝[1], 王怀民[1]

[1](国防科技大学 并行分布国防科技重点实验室,湖南 长沙 410073)

通讯作者: 王涛, E-mail: taowang2005@nudt.edu.cn

摘 要: 代码克隆技术的应用加速了代码搜索,提升了软件复用效率,辅助软件质量评估与恶意代码检测.但代码克隆的应用也为软件开发引入了软件质量问题,提升了软件维护开销的成本.作为软件工程领域的重要研究课题,代码克隆被学术界广泛探索和研究,涌现出了针对代码克隆各个子研究领域的相关文章,包括:代码克隆检索、代码克隆演化、代码克隆分析等.但相关研究中缺少对代码克隆整个领域的全面探索,以及对各个子研究领域发展态势的分析.本文对近十年来代码克隆领域的研究进展进行了梳理和总结.主要包括:(1)对代码克隆子研究领域进行了归纳分类,探究了各子研究领域的整体热度与相关性;(2)分析了代码克隆整体与各子研究领域的发展态势;(3)对比分析了工业界与学术界对代码克隆研究领域的关注情况与变化趋势;(4)从研究人员角度出发,构建了合作关系网络,挖掘出代码克隆领域重要贡献者;(5)统计分析了热门会议、期刊列表.未来代码克隆的热门研究方向包括代码克隆可视化、代码克隆管理等,对于作为支撑技术的代码克隆检测子领域,可以尝试优化方法的可扩展性和执行效率,或从面向特殊克隆检测任务、上下文环境的克隆检测入手进行优化,或将该技术持续应用于其他相关研究领域.

关键词: 代码克隆;系统文献综述;子领域分类;发展态势分析

中图法分类号: TP311

中文引用格式: 张迅晖,王涛,余跃,钟岩,张晏芝,王怀民.代码克隆相关研究综述.软件学报.

英文引用格式: Zhang XH, Wang T, Yu Y, Zhong Y, Zhang YZ, Wang HM. Code Clone Related Studies: A Literature Review. Ruan Jian Xue Bao/Journal of Software, 2021 (in Chinese).

Code Clone Related Studies: A Literature Review

Zhang Xun-Hui[1], Wang Tao[1], Yu Yue[1], Zhong Yan[1], Zhang Yan-Zhi[1], Wang Huai-Min[1]

[1](Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha 410073, China)

Abstract: The application of code clone technology accelerates code search, improves code reuse efficiency, and assists in software quality assessment and code vulnerability detection. However, the application of code clones also introduces software quality issues and increases the cost of software maintenance. As an important research field in software engineering, code clone has been extensively explored and studied by researchers, and related studies on various sub-research fields have emerged, including code clone detection, code clone evolution, code clone analysis, etc. However, there lacks a comprehensive exploration of the entire field of code clone, as well as an analysis of the trend of each sub-research field. This paper collects related work of code clones in the past ten years. In summary, the contributions of this paper mainly include: (1) summarize and classify the sub-research fields of code clone, and explore the relative popularity and relation of these sub-research fields; (2) analyze the overall research trend of code clone and each sub-research field; (3) compare and analyze the

* 基金项目: 科技创新 2030 新一代人工智能重大项目(2018AAA0102304); 国防科技大学重点实验室开放项目(6142110200401)
Foundation item: Science and Technology Innovation 2030 of China (2018AAA0102304); Parallel and Distributed Processing Laboratory (National University of Defense Technology)开放项目(6142110200401)

收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00
CNKI 在线出版时间: 0000-00-00

difference between academy and industry regarding code clone research; (4) construct a network of researchers, and excavate the major contributors in code clone research field; (5) The list of popular conferences and journals was statistically analyzed. The popular research directions in the future include clone visualization, clone management, etc. For the clone detection technique, researchers can optimize the scalability and execution efficiency of the method, targeting particular clone detection tasks and contextual environments, or apply the technology to other related research fields continuously.

Key words: code clone; systematic literature review; sub-research area classification; development trend analysis

1 引 言

克隆代码指代码库中两个及以上相同或相似的源代码片段[1].作为软件工程重要研究课题,代码克隆提升了软件开发效率,辅助软件质量评估以及软件漏洞发现.然而克隆代码作为一类代码异味(bad smell),其复制粘贴的方式会导致缺陷的引入[2],进而增加软件维护成本,导致软件质量的降低,同时代码克隆会引入知识产权保护问题[3].因此,工业界与学术界均密切关注代码克隆问题.

目前存在大量关于代码克隆的相关工作,主要包括代码克隆检测方法研究[4], [5]、代码克隆演化分析[6], [7]、代码克隆可视化[8], [9]、代码克隆重构[10], [11]等.针对上述研究问题,涌现出了大量文献综述,但这些综述类工作主要针对代码克隆研究方法[12]、代码克隆演化[13]、代码克隆可视化[14]这样其中某个子研究领域,但是针对整个代码克隆领域还缺少综述工作科学的对代码克隆子研究领域进行总结,缺少对各子领域的发展态势分析.

针对上述问题,本文将对代码克隆领域近十年的工作进行全方位的搜集整理,抽取文章的基本信息(包括文章发表日期、标题、摘要、关键词、作者及所属单位信息等),通过卡片分类法(card sorting)对文章进行子研究领域划分,最终通过统计方法分析代码克隆各子领域的发展情况以及工业界、学术界对代码克隆的关注程度,并从研究人员角度构建作者合作关系网络,探究代码克隆研究模式,找到为代码克隆领域研究做出了突出研究贡献的研究人员及所属机构,分析国家间研究人员合作的区别.

本文的主要贡献如下:

(1) 对近十年的代码克隆相关文章进行了搜集整理,搜集到 2011-2020 年共 1,294 篇代码克隆相关文章;
(2) 对代码克隆进行了相关子研究领域的人工分类,形成了公开数据集,包含子研究领域的主题、相关文章信息;
(3) 对代码克隆子研究领域进行了热度分析,探究了代码克隆整体及各子研究领域近十年的变化态势;
(4) 对比分析了工业界与学术界对代码克隆各子研究领域的关注程度以及整体关注度的变化;
(5) 构建了代码克隆研究人员合作关系网络,分析了合作关系特征,挖掘了突出贡献人员及热门研究机构,探讨了不同国家合作关系的区别.

本文第 2 节介绍了研究背景,总结了代码克隆相关的系统文献综述文章以及本文与相关工作的区别,提出了本文的研究问题.第 3 节介绍了基于系统文献综述的代码克隆相关文章搜集、筛选、数据提取方法.第 4 节阐述了基于卡片分类的代码克隆子研究领域的分类方法.第 5 节介绍了代码克隆子研究领域分类结果、各类别的发展趋势,工业界学术界的关注情况,作者合作网络分析以及热门期刊、会议分析.第 6 节对文章的结论进行了讨论并给出了代码克隆领域未来研究的展望.最后第 7 节对本文进行了总结.

2 研究背景

虽然已经有很多在代码克隆领域的综述性工作,但这些工作大都专注于代码克隆的某个子研究领域, 表 1 对代码克隆相关系统文献综述进行了总结,对多个方面(“比较项”)进行了比较,包括发表时间、包含文章数量和文章涵盖年份这些“基本信息”,以及“研究点”(指文章关注的代码克隆领域的问题,这里的问题并非“子研究领域”,而是领域中的具体细节问题,例如:“代码中间表示”、“克隆粒度”、“克隆类型”等均属于“代码克隆检测”这一子研究领域)..通过对比发现,过去的系统文献综述工作主要关注于代码克隆的某一个研究领域以及

领域中的部分相关研究点,例如:Ain 等人[15]关注于代码克隆检测方法以及方法中涉及到的相关数据、数据中间表示方式等.虽然有相关工作关注子领域的分类研究和热度分析[16]–[21],但这些工作要么是聚焦在代码克隆某个子领域的具体研究目标,例如:软件相似性的具体研究领域[16],要么在研究工作后经过了长时间的发展,研究领域拓宽且热度可能发生变化[17], [18], [20], [21],要么缺少对子研究领域的发展态势分析[19].同时,我们发现相关研究对代码克隆的分类基于研究经验,分类结果有较大差别,缺少基于科学研究方法的对代码克隆子研究领域的分类.最后,所有的相关研究中缺少对代码克隆在实践领域的关注热度分析.

Table 1 Collection of basic information and focus of code clone related systematic literature reviews

表 1 代码克隆系统文献综述相关工作基本信息与研究点统计

比较项(基本信息和研究点)	[16]	[13]	[17]	[22]	[15]	[23]	[24]	[25]	[26]	[27]	[28]	[14]	[18]	[29]	[19]	[20]	[30]	[31]	[32]	[21]
发表时间	2020	2013	2013	2014	2019	2017	2016	2014	2020	2017	2014	2020	2012	2019	2020	2012	2019	2020	2014	2018
文章数量	136	30	213	7	54	61	30	20	198	177	65	68	262	32	27	220	54	97	39	575
调研文章年份	2002-2019	-2011	1997-2011	-2012	2013-2018	1997-2016	1996-2015	2010-2014	2011-2017	2011-	-2014	-2020	1994-2011	2001-2018	1998-2017	2007-2011	2013-2018	1998-2017	-2014	-2013
关注子研究领域	√		√										√		√	√				√
关注代码克隆相关数据	√	√			√									√		√				
关注克隆检测方法		√	√	√	√			√		√	√		√	√	√	√	√			√
关注代码中间表示					√												√			√
关注增量检测													√				√			
关注克隆粒度				√									√					√		
关注使用工具				√			√											√		
关注相关开源工具			√																	
关注工具依赖性																		√		
关注克隆类型			√	√	√					√			√					√	√	√
关注相关编程语言			√	√	√													√		
关注工具实现				√																
关注集成开发环境													√							
关注检测方法使用的度量							√													
关注方法评估的度量				√					√	√										
关注相似性检测方法	√				√		√													
关注克隆映射方法						√														
关注克隆演化模式		√						√												√
关注克隆遗传图谱抽取方法													√							
关注图形化界面支持																		√		
关注可视化方法												√	√							√
关注重构模式																		√		√

✓ 表示文章关注了该领域

鉴于上述原因,本文提出了如下研究问题:

问题一:代码克隆存在哪些子研究领域?

代码克隆作为热门研究领域被大量工业界、学术界研究人员所关注,但对于代码克隆子研究领域的分类研

究缺少基于科学研究方法的分类结果以及相关研究主题的挖掘.我们通过卡片分类法将近十年国内外代码克隆相关研究进行分类归纳,得到子研究领域与相关主题,并将文章分类结果以开放数据集的形式公开.该问题的解决形成了对代码克隆领域的工作总结,方便了后续代码克隆研究工作的推进.

问题二:代码克隆整体以及各子研究领域呈现何种发展态势?

代码克隆领域的研究经历了长时间的发展,涵盖了包括检测技术的改进、维护软件质量、提升软件开发效率等各个方面的研究.探究代码克隆整体和各个子研究领域发展态势可以帮助后续研究人员把握各领域的发展现状及未来发展趋势,进而有针对性地开展后续研究.

问题三:工业界与学术界对代码克隆的关注情况有何区别?

代码克隆作为软件工程实践相关问题,一直以来被工业界广泛关注,了解工业界对代码克隆的关注情况及关注度变化可以让后续研究人员了解代码克隆在应用实践领域的发展情况,明确软件开发过程中开发者关心的代码克隆问题,进而形成理论与实践的紧密结合.

问题四:代码克隆领域文章作者以何种形式进行研究合作?

从研究者角度出发,构建代码克隆领域研究人员合作网络,探究活跃研究人员、研究团队,挖掘整体和不同国家研究人员合作模式可以帮助后续研究人员跟踪热门研究者、研究团队的相关研究工作,形成相关研究知识脉络,构建合作关系.

问题五:代码克隆领域文章倾向于发表在哪些期刊、会议中?

从研究者角度出发,通过对热门会议、期刊的分析,可以帮助后续研究人员有针对性的进行论文投稿,同时可以参与、跟踪相关研究会议、期刊的工作,加速科研进程.

3 系统文献综述

本文遵循软件工程领域系统文献综述的规范步骤[33],主要包括五个步骤: 在线查找,文章筛选,迭代滚雪球(recursive snowballing),质量评估与数据抽取(如图 1 所示).

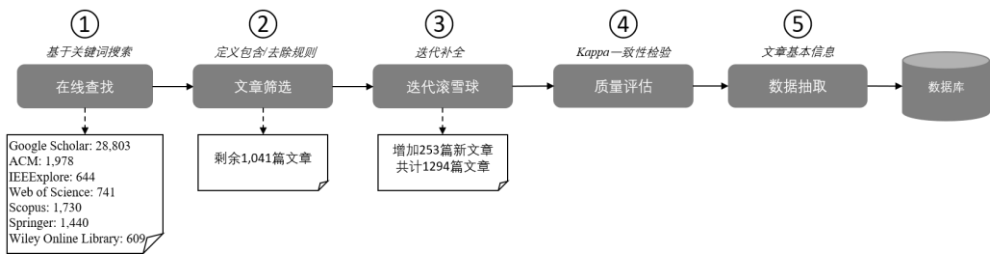

Fig.1 Pipeline of systematic literature review

图 1 系统文献综述流程

3.1 在线查找

3.1.1 选择检索方法

目前系统文献综述的在线查找方法主要分为两种:基于关键词搜索[33]和基于目标会议期刊搜索[34],两种搜索策略都存在各自的弊端.基于关键词搜索主要依赖于关键词和搜索引擎的选取,其中搜索关键词的形成主要依赖于研究问题以及作者的经验[13], [35], [36],会出现关键词同义词漏掉的情况,同时检索数据库和出版商的选择上也会漏掉一些相关文章.对于基于目标会议期刊的搜索,目标的选取是关键,而目标期刊会议的缺失会导致大量相关文献的丢失.虽然迭代滚雪球方法可以帮助减少丢失文章数量[37],但仍然无法保证数据的完整性.

鉴于上述因素,我们最终选择了基于关键词搜索的方法,主要原因如下:

(1) Sobrinho 等人[34]使用了基于目标会议期刊搜索的方法,但在迭代滚雪球阶段,他们新发现的 85 篇文章出现在了 60 个未覆盖到的期刊会议中;

(2) 从 WikiCFP[1]中我们发现,仅计算机科学类别下就涵盖了数千个期刊会议,因此有效选择目标期刊会议实现文章全集的覆盖非常困难;

(3) 对于基于关键词搜索的方法,我们可以综合相关工作中作者定义的关键词和选择的检索数据库来优化我们的检索策略.

3.1.2 定义检索字符串

为了避免检索关键词的丢失,我们搜集了部分相关工作中的检索关键词,结果如表 2 所示.我们可以将检索关键词分为两个部分:

(1) 主体: ‘code’, ‘software’, ‘application’

(2) 动作: ‘clone’, ‘cloning’, ‘copy’, ‘duplicate’, ‘duplication’, ‘similarity’, ‘sibling’

因此,我们将主体和动作中的关键词进行组合并用或(OR)逻辑进行连接,最终形成了如下所示的检索字符串:

“code clone” OR “code cloning” OR “code copy” OR “code duplicate” OR “code duplication” OR “code similarity” OR “code sibling” OR “software clone” OR “software cloning” OR “software copy” OR “software duplicate” OR “software duplication” OR “software similarity” OR “software sibling” OR “application clone” OR “application cloning” OR “application copy” OR “application duplicate” OR “application duplication” OR “application similarity” OR “application sibling”

Table 2 Search keywords or string in related papers

表 2 相关工作中的检索关键词或字符串

文章	检索关键词或字符串
[17]	clone; software; code
[22]	code clone; code clone tools; code clone prevention; code clone prevention mechanism; code clone management
[13]	((‘code’ OR ‘software’ OR ‘application’) AND (‘clone’ OR ‘cloning’ OR ‘copy’ OR ‘duplicate’ OR ‘duplication’ OR ‘similarity’) AND (‘change’ OR ‘evolution’ OR ‘genealogy’ OR ‘maintenance’ OR ‘management’ OR ‘tracking’))
[38]	code; software; clone; clones; duplication
[30]	code clone detection; text-based techniques; tree-based techniques; metric based techniques; PDG based techniques; hybrid techniques; machine learning techniques
[25]	((‘code cloning’ OR ‘software code cloning’ OR ‘code cloning tool’) AND (‘code copy’ OR ‘duplicate code’ OR ‘duplication’ OR ‘code similarity’))
[34]	code siblings; copy-and-paste; duplicate code; near-miss clones

3.1.3 选择检索数据库

综合相关工作中对检索数据库和搜索引擎的选择[17], [33], [35], [39],本文中检索的数据库包括: ACM, IEEExplore, Scopus of Elsevier, Springer, Web of Science, Google Scholar, Wiley Online Library.

3.1.4 检索工具和方法

为了快速准确的检索不同线上资源,我们使用了多种辅助工具.对于 ACM, Springer 和 Web of Science,我们

[1] http://www.wikicfp.com/cfp/

使用了 GitHub 中的开源 Chrome 浏览器插件 lit-automation/chrome-plugin.[1]对于 Google Scholar,我们使用了 Publish or Perish[40].对于 IEEExplore, Wiley Online Library 和 Scopus,我们通过人工检索的方式在官方网站中搜索查询字符串.

对于 Google Scholar, IEEExplore 以及 Scopus, 他们都有最大检索字符串长度的限制,因此我们将检索字符串分为如下两部分分别检索:

(1) 子串 1: *"code clone" OR "code cloning" OR "code copy" OR "code duplicate" OR "code duplication" OR "code similarity" OR "code sibling" OR "software clone" OR "software cloning" OR "software copy" OR "software duplicate" OR "software duplication".*

(2) 子串 2: *"software similarity" OR "software sibling" OR "application clone" OR "application cloning" OR "application copy" OR "application duplicate" OR "application duplication" OR "application similarity" OR "application sibling".*

对于 Google Scholar 的检索,一次最多返回 1000 个结果[41], [42],为了可以覆盖所有相关文章,我们按照年份对检索过程进行拆分,保证每次检索结果少于 1000,如果依然不能满足要求,我们继续对检索字符串进行拆分直到满足要求为止(例如:在使用子串 1 搜索 2020 年 Google Scholar 结果的时候返回结果超过了 1000,因此我们将子串 1 又拆分成了 2 个字符串).

3.1.5 形成检索结果

通过上述检索策略我们共搜集到了 35,945 篇文章(检索时间为:2020-09-13),其中 Google Scholar (28,803), ACM (1,978), IEEExplore (644), Web of Science (741), Scopus (1,730), Springer (1,440), Wiley Online Library (609).

3.2 文章筛选

本章我们设置了文章筛选的包含和去除规则,然后通过人工筛查的方式对文章进行筛选.这里我们设置的规则和遵循的步骤如下:

(1) 去重.不同的搜索引擎和检索数据库可能会检索到一篇文章,这里我们首先根据文章题目删除重复文章(7,901 篇文章被删除);

(2) 删除不相关文章.我们基于文章标题、摘要、关键词以及全文信息,通过人工检查的方式删除了与代码克隆无关的文章(24,982 篇文章被删除);

(3) 删除非中、英文文章(523 篇文章被删除);

(4) 删除非学术研究论文.我们仅保留了学术研究类工作,删除了例如专利、会议介绍、会议幻灯片等工作(1,101 篇文章被删除);

(5) 删除无法找到全文的工作.为了后续的数据抽取,我们删除了无法找到全文的工作(51 篇文章被删除);

(6) 保留了 2011-2020 年之间的工作.本文仅调研代码克隆近十年的发展情况(346 篇文章被删除).

通过上述步骤的筛选,共剩余 1,041 篇文章.

3.3 迭代滚雪球

由于基于关键词的检索方法在构建检索字符串的时候存在潜在的同义词缺失问题,为了解决该问题,我们利用迭代滚雪球的方法在相关文章的参考文献中检索丢失的相关文章[37].相比于非迭代滚雪球方法[43],该方法可以获得更多的相关文章从而规避文章丢失的问题.

迭代滚雪球的每个阶段主要包括两个步骤:

(1) 自动抽取参考文献.这里我们使用 CERMINE[44]工具自动抽取文章的参考文献列表;

(2) 根据 3.2 中定义的文章筛选规则确定文章是否相关.

基于上述步骤,经过 5 次迭代后没有新的相关文章出现.每次迭代的文章获取和筛选情况如表 3 所示.

[1] https://github.com/lit-automation/chrome-plugin

3.4 质量评估

在本文中第一和第四作者负责了所有的人工检查工作,涵盖了第一轮检索后和迭代滚雪球两个阶段的文章筛选工作,两名作者均具备 5 年以上的软件工程开发经验,并且在人工检查开始前对代码克隆的基本定义及相关文章涵盖的范围进行了讨论和分析.为了验证两位作者在文章筛选过程中筛选标准的一致性,我们利用卡帕检验(Kappa)的方法进行了分析.我们在去除了重复文章后随机选取了 100 篇文章,要求两位作者按照 3.2 定义的文章筛选规则分别评判文章是否应该保留,之后计算两位作者的卡帕一致性系数(κ)[45].结果表明两位作者在文章筛选阶段的评价标准具有近乎完美的一致性($\kappa = 0.928$)[46].

Table 3 The selection of papers for each stage during recursive snowballing

表 3 迭代滚雪球方法每阶段文章筛选情况

	阶段					
	1	**2**	**3**	**4**	**5**	**总计**
抽取文章数量(-)	21797	4461	535	41	6	26840
重复文章数量(-)	15289	3100	339	26	5	18759
不相关文章数(-)	5217	1112	166	7	0	6502
非中、英文文章数(-)	20	1	0	0	0	21
非研究论文文章数(-)	45	10	0	0	0	55
未找到全文文章数(-)	74	9	1	0	0	84
2011 年之前发表文章数(-)	928	203	27	7	1	1166
剩余相关新文章	224	26	2	1	0	253
注解	数字表示文章数量					
	(-)表示删除文章操作					

迭代滚雪球之后,我们获取了 253 篇新文章.综合之前的文章,共搜集到代码克隆相关文章 1294 篇.

3.5 数据抽取

为辅助后续文章主题抽取、子研究领域分类、热度分析、工业界学术界研究分析以及作者合作网络分析,我们分别对文章信息及文章相关作者信息进行了提取,提取的数据包括如下内容:

(1) 文章信息:会议期刊名称、发表时间、标题、摘要、关键词;

(2) 作者信息:作者姓名、文章发表时所属单位、文章发表时所属国家、文章发表时邮件地址、排序.

4 代码克隆子研究领域分类方法

Table 4 Classification of sub research areas of code clone in related work

表 4 相关工作中代码克隆子研究领域分类结果

文章	**子研究领域**
[17]	代码克隆演化;代码克隆分析;代码克隆对软件质量的影响;网站中的代码克隆检测;代码克隆相关领域;面向方面编程中的代码克隆检测
[18]	代码克隆分析;代码克隆检测;代码克隆管理;代码克隆工具评估
[19]	代码克隆管理;代码克隆检测;代码克隆可视化;代码克隆重构;代码克隆追踪;关联代码克隆编辑;代码克隆与软件质量控制
[20]	代码克隆检测;代码克隆管理;代码克隆演化;代码克隆删除;代码克隆与缺陷;代码克隆可视化;代码克隆评估与检测;代码克隆分类;代码克隆与版本演化;代码抄袭;版权保护;软件生产线;面向方面编程;软件质量分析;代码克隆涌现;代码克隆分析;程序理解;其他
[21]	克隆检测;克隆分析;克隆维护与管理;综述和工具评估

对于代码克隆子研究领域的研究,相关工作得出了不同的分类结果,如表 4 所示.针对分类不一致的问题,本文在综合了近十年的更全面的代码克隆相关研究集合的基础上,利用软件工程科学分类方法——“卡片分类”[47], [48],将代码克隆子研究领域进行归纳整理,并将子研究领域、包含主题以及文章对应关系以数据集的形式发布到 GitLink 平台[1].

本文基于卡片分类方法的代码克隆子研究领域分类主要包括如下几个步骤:

(1) 准备工作:前四位作者共同承担卡片分类系列工作,他们通过首先根据文章的标题、关键词、摘要信息甚至全文为每一篇相关文章添加卡片.卡片上的信息可以根据开发者经验或相关信息自行总结或提取.这里所说的卡片即代码克隆相关主题,卡片信息包括两个部分:主题的英文名称,中文描述信息(便于理解主题,加速后续新文章的合并).

(2) 卡片分类:卡片分类方法主要有两种,一种是分开并行编码后检查分类一致性;另一种是单线程共同分类[49].我们采用了后一种方法,在分类过程中就分歧进行讨论并达成一致[50],该方法可以快速构建领域知识,及时更新认知并形成一致性结论.这里我们使用了开放分类方式(open coding approach)[48],在分类过程中产生子研究领域.作者们按照卡片中的主题描述信息,将关注相同研究领域或类似研究问题的主题进行整合(例如: “license violation detection”这个主题描述为 “利用克隆检测对违反开源许可证的行为进行检测”; “bug localization”这个主题描述为 “利用代码克隆技术对代码中的错误进行定位”.这两个主题均为代码克隆检测技术在其他领域的应用,因此进行合并,并用卡片“other fields based on clone detection technique”代替.最终整合得到的所有一级卡片即所有子研究领域,而包含的所有初始卡片即该子研究领域包含的主题).

5 结果分析

本章将对文章的结果进行分析和总结,回答第 2 节中提出的 5 个研究问题.

5.1 问题一:代码克隆存在哪些子研究领域?

5.1.1 子研究领域整体热度分析

通过卡片分类方法,我们得到了如表 5 所示的分类结果.其中*代码克隆管理*属于一个高维类别,根据 Bharti 等人[51]的介绍,代码克隆管理涵盖了代码克隆检索、代码克隆可视化、代码克隆重构、代码克隆检测等各个领域,因此本文在进行分类时只将提出克隆管理相关框架的相关文章整合到了*代码克隆管理*类别下.为了便于后续研究,我们单独设置了*代码克隆调研*类别,该类别与其他类别存在交集,即*克隆调研*相关文章同属于其他研究子领域.

从表 5 中我们可以看出,代码克隆检测在近 10 年的整体情况来看是最热门的研究领域,这是因为该领域是其他研究领域的基础,包括代码克隆分析,代码克隆演化等都依赖于成熟的代码克隆检测技术.虽然代码克隆检测依赖于大量的验证数据集和验证方法,但相对来说代码克隆数据集和代码克隆评估这两个研究领域的相关文章较少,大量克隆检测算法的研究依赖于相同的大规模数据集和验证算法.基于代码克隆检测算法,派生出了大量分析类工作和克隆演化与克隆重构两个分支领域,这些领域主要分析和解决软件或软件社区质量与健康度的问题.同时我们可以发现,代码克隆检测技术渗透到了大量其他的软件相关领域,有 90 篇研究工作关注于基于克隆检测技术的其他研究领域的相关研究.

[1] https://www.gitlink.org.cn/Nigel/jos_code_clone_trend_future/

Table 5 Classification of sub research areas of code clone

表 5 代码克隆子研究领域分类结果

类别	文章数量	包含文章描述
代码克隆检测(clone detection technique)	749	相关文章提出了新的克隆检测方法
代码克隆分析(clone analysis)	183	相关文章利用克隆检测方法或针对代码克隆领域进行了实证研究（不包含专门针对其他子研究领域的分析类文章）
代码克隆演化(clone evolution)	143	相关文章关注了软件生命周期中代码克隆的变化情况
代码克隆重构(clone refactoring)	110	相关文章关注于克隆代码的重构问题
基于克隆检测技术的其他研究领域 (other fields based on clone detection technique)	90	相关文章利用克隆检测技术解决其他研究领域问题
代码克隆调研(survey and tutorial)	87	相关文章对之前领域的相关文章或技术报告进行了汇总和分析
代码克隆可视化(clone visualization)	52	相关文章主要贡献为克隆的可视化方法或工具的提出
代码克隆评估(clone evaluation)	28	相关文章介绍了评估代码克隆检测方法的指标、工具、方法等
代码克隆数据集(benchmark)	21	相关文章构造或评估了关于代码克隆检测的数据集
代码克隆管理(clone management)	14	相关文章关注于代码克隆管理的宏观层面（例如:提出框架、机制、概念等）

5.1.2 子研究领域相关性分析

在我们的分类中,一篇文章可以同属于多个类别,图 2 展示了不同子研究领域相关文章的交集情况(这里为了便于展示,图中只包含了含有最少 2 篇文章的集合相交情况,完整图片已上传 GitLink[1]).从图中我们看出,*代码克隆调研*相关文章主要集中在对代码克隆检测技术的总结(87 篇文章中有 65 篇涉及到对克隆检测方法的讨论),而对于其他的子研究领域来说,除了*基于克隆检测技术的其他研究领域*外都存在相关的调研报告,本文将在 5.1.3 节对该领域的相关研究主题进行分析,对*基于克隆检测技术的其他研究领域*进行详细的总结阐述.另外我们发现,*代码克隆演化*分别与*代码克隆可视化*、*代码克隆分析*以及*代码克隆重构*存在较强的相关性,分别有超过 10 篇的相关文章,这说明基于代码克隆演化的可视化支持以及面向软件质量的克隆分析与重构是*代码克隆演化*的主要研究方向.

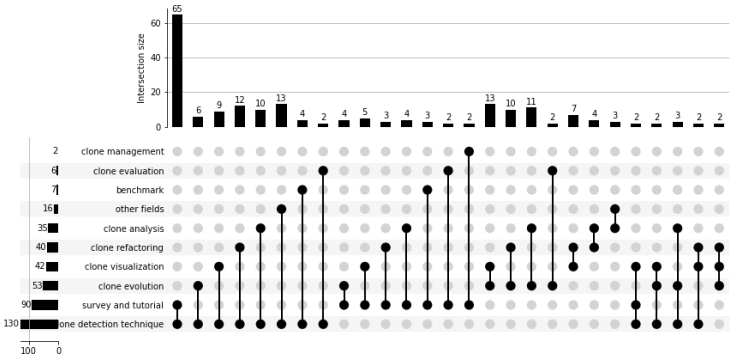

Fig.2 Intersection plot of different sub research fields

图 2 不同子研究领域集合交集图

1 https://www.gitlink.org.cn/Nigel/jos_code_clone_trend_future/tree/master/upsetplot.png

5.1.3 子研究领域关联主题分析

在卡片分类过程中,我们对相关研究设置了所属主题,然后对主题进行了整合,最终形成了子研究领域与所属主题的对应关系(具体流程见第 4 章中的卡片分类步骤描述).图 3 展示了*基于克隆检测技术的其他研究领域*的相关研究主题,由于相关调研工作中缺少对该领域的详细分析,这里我们对该研究领域进行详细介绍(所有研究领域的主题关联图已以树状图的形式上传 GitLink[1]).

通过统计发现,所有的 90 篇*基于克隆检测技术的其他研究领域*的文章中,共分类产生了 41 个主题,其中代码检索(code search)(19 篇相关文章)、恶意软件检测(malware detection) (13 篇相关文章)以及漏洞检测(vulnerability detection) (10 篇相关文章)是最热门的三个主题.由于代码克隆检测的特点,在软件工程领域,代码克隆本身可以作为一类方法,通过计算代码之间的相似性,辅助代码复用以及软件质量管理等.因此,代码克隆本身存在优势和弊端,他虽然会增加代码维护成本[52],导致软件漏洞的传播[53],降低代码的易读性[54]等,但同时代码克隆技术还可以提升软件复用效率[55],加快软件开发速度[56],发现软件漏洞[57]以及预测代码错误[58]等.

通过对子研究领域包含主题的统计和分析可以帮助后续研究人员快速建立领域知识,结合我们构建的相关文章数据集达到快速入门、构建完整知识脉络的目的.

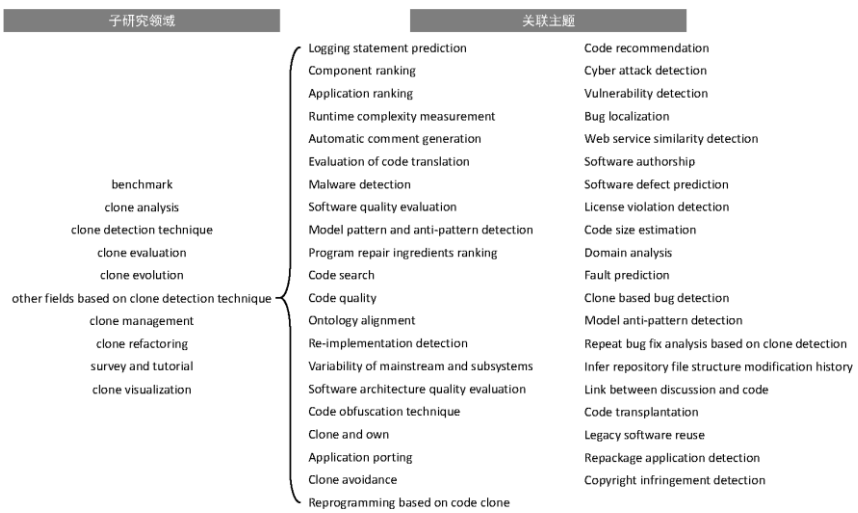

Fig.3 Related topics of other fields based on clone detection technique

图 3 基于克隆检测技术的其他研究领域关联主题

5.2 问题二:代码克隆整体以及各子研究领域呈现何种发展态势?

我们按照时间顺序对克隆检测整体和各子研究领域的发展态势进行了统计分析,结果如图 4、图 5 所示(图片显示代码见 GitLink[23]).从整体的热度变化情况来看,代码克隆从 11 年到 12 年经历了快速发展达到了研究热度的高峰,后续的研究热度呈现曲折下降的发展态势.由于相关文章的检索在 2020 年 9 月开展,我们没有拿到 2020 年的完整数据,我们暂时不考虑 2020 年代码克隆相关文章热度.从子研究领域我们可以发现,*代码克隆检*

1 https://www.gitlink.org.cn/Nigel/jos_code_clone_trend_future/tree/master/class_topic_relation_tree.html

2 https://www.gitlink.org.cn/Nigel/jos_code_clone_trend_future/tree/master/overall_hotness_change_trend.html

3 https://www.gitlink.org.cn/Nigel/jos_code_clone_trend_future/tree/master/sub-fields_hotness_change_trend.html

*测*的热度变化趋势与整体变化趋势相近,这可能是由于*代码克隆检测*相关文章在所有文章数量中占了巨大的比重.

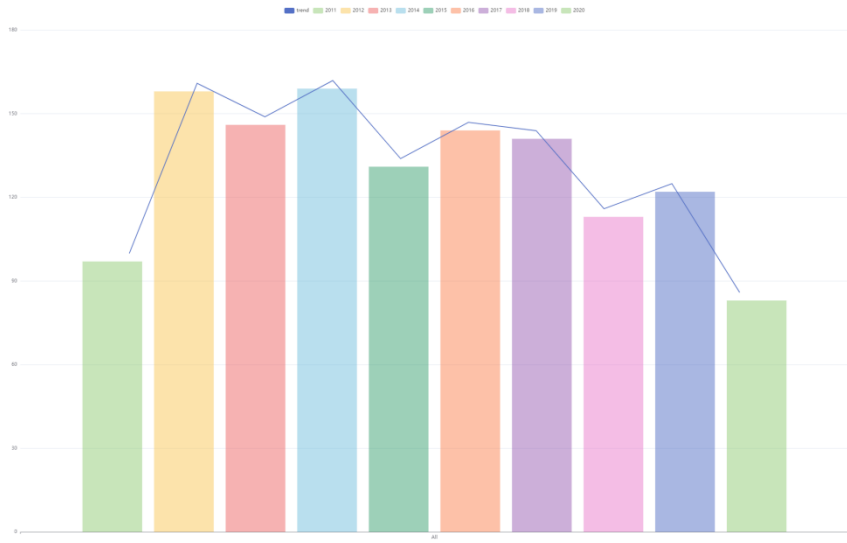

Fig.4 The trend of the overall perspective of code clone

图 4 代码克隆整体研究热度变化趋势

Fig.5 The trend of sub-research fields of code clone

图 5 代码克隆子研究领域热度变化趋势

对比其他子研究领域我们发现,包括*代码克隆分析*、*代码克隆演化*以及*代码克隆重构*在内,2015 年相关文章的数量都大幅度下降(趋势与*代码克隆检测*相反),然而在接下来的一年到两年时间内热度有了一定程度的提升,可能的原因是由于克隆检测方法是其他子研究领域的基础,15 年新代码克隆检测方法的提出促进了后续代码克隆其他子研究领域的发展.但从整体趋势来看,这三个子研究领域与*代码克隆检测*的发展趋势相近,呈现整体上下降的趋势.

从整体发展趋势来看,*代码克隆可视化*、*代码克隆数据集*、*代码克隆管理*三个子研究领域在近一两年维持了一个稳定或者上升的态势,没有因为整体发展态势的变化而下降,我们认为可能的原因如下:

(1) 数据集是代码克隆检测方法形成和突破的基础,其发展会由于克隆检测方法出现瓶颈而有所突破.新数据集的提出可能意味着代码克隆检测方法将在某种语言或某个方面有重大进步;

(2) 代码克隆可视化和代码克隆管理框架的提出是对代码克隆检测方法在软件维护和管理方面的应用,相关工作的发展相较于代码克隆检测方法具有一定的滞后性.

5.3 问题三:工业界与学术界对代码克隆的关注情况有何区别?

对于工业界与学术界对代码克隆研究的关注情况,我们根据抽取到的作者所属单位信息,通过人工审查的方式进行了归类,我们将包含了隶属工业界作者的文章认定为工业界关注的文章,如果只包含学术研究机构则认定为仅学术界关注的文章.

统计发现,近 10 年代码克隆相关文章中,工业界参与的文章包含 112 篇(8.66%),说明工业界对代码克隆领域有一定关注度.我们对比了每个子研究领域的相关文章占比情况(如图 6 所示).从图中我们可以看到,工业界对*代码克隆管理*、*代码克隆评估*、*代码克隆重构*以及*代码克隆分析*这几个子研究领域相对更加关注(文章数量所占百分比更高),而对于*代码克隆检测*领域来说,关注热度反而相对较低.由此可见,工业界相对更关注于代码克隆对软件质量带来的影响以及如何重构和管理代码克隆.

对比工业界与学术界对于代码克隆研究热度随时间的变化情况(如图 7 所示),我们发现相比于学术界(2014 年),工业界(2012 年)更快达到了研究热度的高峰,并且工业界在最近 2 到 3 年对代码克隆的研究呈现出一种回温的态势,而学术界在整体的研究热度上存在下滑趋势.对于该现象可能的原因是,代码克隆的相关问题是由工业界从实际问题出发而发现得来,之后学术界关注该问题并持续提出新思路、新方法、新结果,进而在工业界中予以采纳和应用.因此工业界对代码克隆相关问题持续关注.

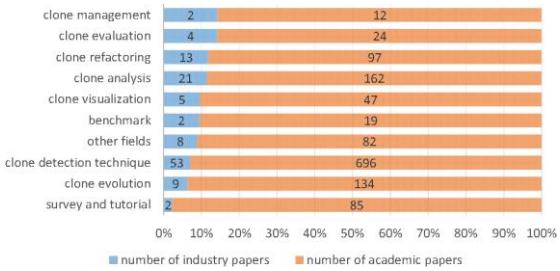

Fig.6 Comparison between industry and academic research hotness of code clone

图 6 代码克隆子研究领域工业界与学术界研究热度对比分析

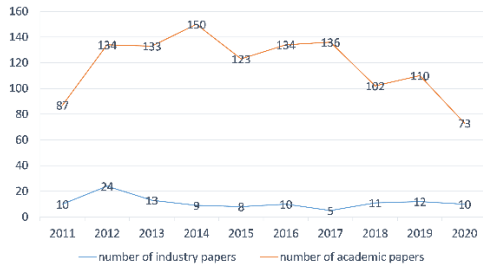

Fig.7 Change of number of industry and academic research papers overtime

图 7 代码克隆工业界与学术界研究相关文章数量随时间的变化

5.4 问题四:代码克隆领域文章作者以何种形式进行研究合作?

图 8 展示了代码克隆领域作者的合作网络(矢量图见 GitLink[1]),其中节点表示作者,边表示两名作者共同合作过至少一篇文章,节点大小与作者参与文章数量正相关,节点对应的作者姓名大小与节点大小正相关,颜色表示作者所属的国家和地区(同一颜色表示相同国家或地区),边的颜色深度与相关作者合著文章数量正相关.

经统计发现,有 72.4% (1,568)的作者仅参与过 1 篇文章,4.9% (106)的作者参与撰写了 5 篇及以上数量的代码克隆相关文章,其中参与数量最多的作者为 Chanchal K. Roy (如图 8 所示面积最大的节点),该作者共参与撰写了 95 篇代码克隆相关文章.因此虽然代码克隆领域属于热门研究领域,有大量研究人员参与了相关子领域的研究工作,但实际上只有较少的研究人员专注于该领域的研究并持续贡献研究成果.我们对作者合作关系网络特征进行了分析,发现整个网络的平均聚类系数(C)为 0.913[59],平均路径长度(L)为 4.906[60],结合 Telesford 等人[61]提出的鉴别小世界复杂网络的方法,我们使用 Gephi[62],根据连线概率生成了等效随机衍生网络,得到该随机网络的平均聚类系数(C_{rand})为 0.001,平均路径长度(L_{rand})为 6.332.在此基础上我们计算作者合作网络的小世界系数($\sigma = \frac{C/C_{rand}}{L/L_{rand}}$=1178.38)[63], [64].综上所述,代码克隆作者合作网络基本满足小世界网络的特征,即 $C \gg C_{rand}, L \approx L_{rand}, \sigma > 1$.从代码克隆作者合作关系网络整体可以看出,代码克隆的相关研究呈现出“小世界网络”的特点,作者呈现成簇的聚集性.同时从作者的研究频率来看,呈现“高热度、低持续”的特点,即虽然有大量研究人员参与到了代码克隆的相关研究工作中,只有少数研究人员持续关注该领域.

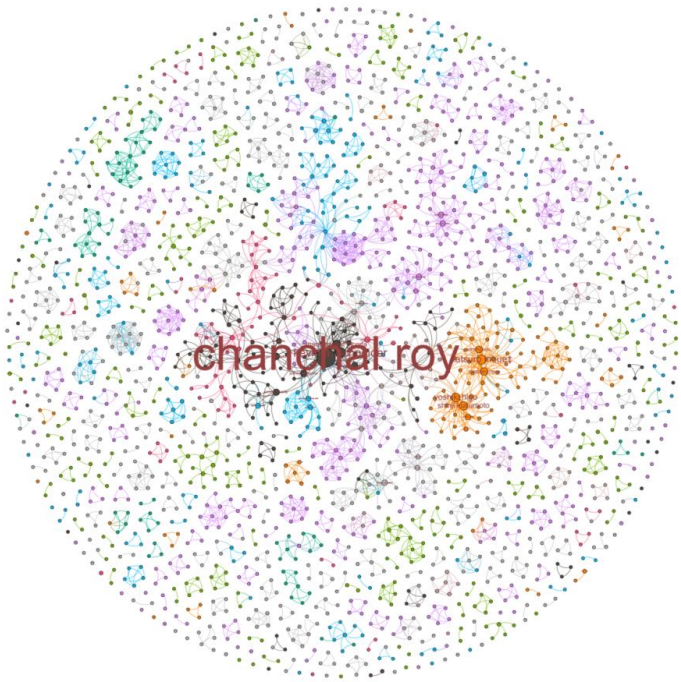

Fig.8 Co-author network of code clone research field
图 8 代码克隆相关研究作者间合作关系网络

从研究机构所属的国家来看,排名靠前的国家分别是中国、印度、美国、加拿大、日本.我们分别对每个国家的作者合作关系网络的节点度和图密度进行统计(见表 6).结合作者的文章产出情况来看(节点大小),来自加

1 https://www.gitlink.org.cn/Nigel/jos_code_clone_trend_future/tree/master/author_network.pdf

拿大萨斯喀彻温大学(University of Saskatchewan)团队(代表研究人员: Chanchal K. Roy, Kevin A. Schneider),以及日本的大阪大学(Osaka University)、南山大学(Nanzan University)联合团队(代表研究人员: Katsuro Inoue, Shinji Kusumoto, Yoshiki Higo, Masami Noro)在代码克隆领域持续关注且做出大量贡献.相比较而言,中、印、美三国各自子图的图密度相对较小,说明作者之间的合作关系不紧密,然而我国子图的平均节点度相较于印度和美国较高,说明我国存在一些高产团队或活跃作者,拉高了整体的平均合作程度.

Table 6 Proportion of authors from research institutions of different countries and analysis of co-author networks

表 6 各国研究机构相关作者占比与关联网络分析

	中国	印度	美国	加拿大	日本
作者占比(%)	24.46	15.64	10.66	6.32	6.05
平均节点度	3.694	1.917	2.476	3.956	4.137
子图密度	0.007	0.006	0.011	0.029	0.032

为了帮助未来研究人员更好地追踪相关子研究领域的热门开发者和开发团队,我们按照发表文章的数量分别统计了每个子研究领域的相关研究人员及所属研究团队(结果如表 7 所示).

表 7 各子研究领域热门研究人员与研究团队整理

子研究领域	热门研究人员[所属研究团队](发表文章数量)
代码克隆检测	Chanchal K. Roy[University of Saskatchewan](25); Yoshiki Higo[Osaka University](13); Oscar Karnalim[University of Newcastle, Maranatha Christian University](12)
代码克隆演化	Chanchal K. Roy[University of Saskatchewan](29); Kevin A. Schneider[University of Saskatchewan](25); Manishankar Mondal[University of Saskatchewan](22)
代码克隆重构	Katsuro Inoue[Osaka University](8); Chanchal K. Roy[University of Saskatchewan](7); Masami Noro[Nanzan University](6)
基于克隆检测技术的其他研究领域	Mohammad Reza Farhadi[Concordia University](5); James R. Cordy[Queen's University](5); Katsuro Inoue[Osaka University](5)
代码克隆调研	Ritu Garg[Indira Gandhi Delhi Technical University for Women](4); Chanchal K. Roy[University of Saskatchewan](3); Dhavleesh Rattan[Punjabi University](2)
代码克隆可视化	Chanchal K. Roy[University of Saskatchewan](7); Kevin A. Schneider[University of Saskatchewan](6); Katsuro Inoue[Osaka University](5)
代码克隆评估	Jeffrey Savjlenko[University of Saskatchewan](12); Chanchal K. Roy[University of Saskatchewan](11); Matthew Stephan[Queen's University, Miami University](4)
代码克隆数据集	Jeffrey Savjlenko[University of Saskatchewan](3); Alan Charpentier[University of Bordeaux](3); Chanchal K. Roy[University of Saskatchewan](3)
代码克隆管理	Minhaz F. Zibran[University of Saskatchewan, University of New Orleans](3); Hamid Abdul Basit[Lahore University of Management Sciences](2); Jan Harder[University of Bremen](2)

从结果可以看出,不同子研究领域的热门研究人员和团队有所区别,但我们发现 University of Saskatchewan 团队和 Osaka University 团队的研究人员在多个子领域都处于积极研究的状态.

5.5 问题五:代码克隆领域文章倾向于发表在哪些期刊、会议中?

图 9 展示了排名前 10 位的热门期刊、会议列表,包括了发表的相关文章数量,文章所占百分比以及对应期刊、会议名称.从结果中我们可以看到大量文章出自软件克隆国际研讨会(International Workshop on Software Clones),该会议是 ICSME（International Conference on Software Maintenance and Evolution）会议下设的研讨会,截至 2021 年已举办了 15 届.后续研究人员在进行论文投稿或相关会议追踪的时候,可以考虑该研讨会.

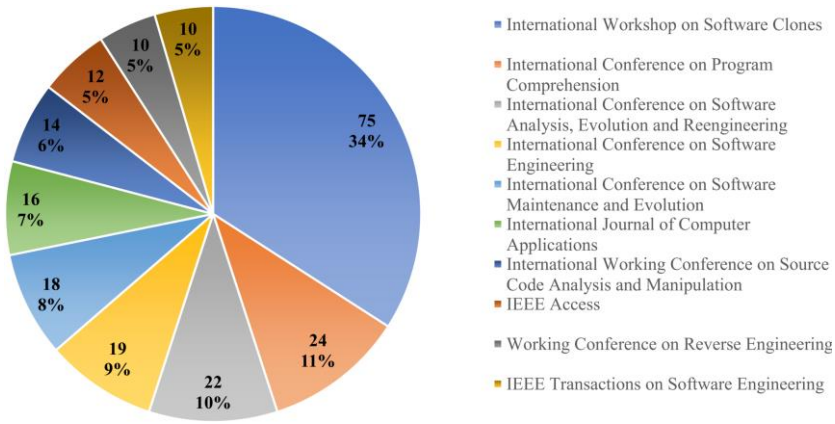

Fig.9 The submission distribution of popular journals and conferences

图 9 代码克隆领域热门期刊、会议投稿情况分布图

6 讨 论

从代码检测的热度分析结果来看,总体上对该领域的研究热度有下降的趋势,但是对于克隆可视化、克隆管理、相关调研分析等有上升的趋势.同时我们发现,工业界自 2017 年以来,对代码克隆相关研究的关注度逐渐提升,而且相对学术届来说更加关注的领域是克隆管理、评估以及重构的工作.未来研究人员在进行科学研究的时候可以从可视化管理的研究热点出发,丰富相关工具、完善相关管理系统,尽可能实现成果转化,将代码克隆管理演技成果应用到相关企业的软件开发过程中.

作为支撑技术,代码克隆检测一直是代码克隆的最热门子研究领域.针对该领域的研究,优化方向包括了准确性、可扩展性和执行效率三个方面[65].关于准确性,随着深度学习技术的引入,现有的克隆检测方法对于检测各种类型(Type 1-4)的克隆都有非常好的效果[66], [67].但实际上深度学习算法在代码克隆领域应用的泛化性问题还未得到充分解决[68],由于 BigCloneBench、OJClone 等数据集的形成仅面向少量的函数或题解,缺少对真实开源项目中克隆代码的覆盖,因此很难直接将训练模型应用到工业生产环境中.对于可扩展性和执行效率,大量研究工作关注于利用 GPU 加速[69]、索引技术[70]–[73]来加速检测速度、过滤候选代码片段.除此之外,很多开发者开始分析现有克隆检测算法在特殊的克隆问题上的表现情况（large gap[74]、large variance 克隆[70]）,并给出对应解决方案.同时大量相关研究关注于对特殊代码的克隆检测和分析,例如:智能合约[75],以及跨语言的克隆检测算法的研究[76], [77].未来研究人员如果希望优化现有的代码克隆检测算法,可以首先考虑如何构建扩展性强的代码克隆数据集,采用什么方法才能不断产生高质量的克隆标注,进而辅助基于深度学习的代码克隆检测方法,提升克隆检测准确性和泛化性.其次,研究人员可以从执行效率和可扩展性方面入手,或者考虑在特殊的上下文环境下优化克隆检测技术.

代码克隆检测作为一项关键技术,在代码推荐、恶意代码检测、软件质量评估等各个领域发挥着重要作用.虽然关于“基于克隆检测技术的其他研究领域”的相关研究热度呈下降趋势,但我们认为涉及基于代码分析、软件复用等的大量相关研究都可以将代码克隆检测作为解决问题的方法或者优化技术.随着开源软件的发展,

软件复用趋势越来越明显,开源世界中充斥着大量的克隆代码[78],这对软件维护带来了不小的挑战.在这种情况下,如何预测并发现可能引发 bug 的克隆[79],如何自动化重构代码克隆[80],如何在开发过程中可视化管理代码克隆[81]都将成为未来的重点研究问题.

7 总 结

代码克隆作为软件工程重要研究领域被研究人员广泛关注,大量相关研究针对代码克隆的各个子研究领域进行深入探索,然而目前缺少针对代码克隆各子研究领域的综合介绍和热度分析.针对该问题,本文按照系统文献综述的详细步骤,搜集了近 10 年来的 1,294 篇关于代码克隆的相关研究工作,通过卡片分类法,将代码克隆的相关研究划分到了 10 个子研究领域,分别探究了:(1)代码克隆子研究领域的整体热度与相关研究的领域相交情况,挖掘出了代码克隆的重点研究方向以及各子研究领域的相关主题;(2)分析了代码克隆整体与各子研究领域的热度随时间的变化情况,发现了整体热度下降的态势以及各子研究领域发展态势的差异;(3)对比分析了工业界与学术界对代码克隆子研究领域的关注度区别以及关注度随时间的变化情况,发现了工业界对代码克隆管理与软件质量维护方面的倾向性;(4)构建了作者合作关系网络,探索出了代码克隆研究的“小世界”、“高热度、低持续”的特点,发现了热门研究者与研究团队,分析出了不同国家研究机构关于代码克隆研究的差异性;(5)统计了热门投稿会议和期刊,对后续研究人员进行论文投稿和相关研究跟踪有一定的指导意义.

本文将搜集到的相关工作以及分类结果以数据集的形式公开,结合本文的研究结果,可以帮助后续研究人员快速构建领域知识,了解相关工作,追踪研究热点以及热门研究人员、团队、会议、期刊.本文从宏观的角度对代码克隆研究领域进行了分析,未来研究工作可以在本文结论和数据的指导下对各子研究领域进行更深入的探究.

References:

[1] 陈秋远, 李善平, 鄢萌, and 夏鑫, “代码克隆检测研究进展,” 软件学报, vol. 030, no. 004, pp. 962–980, 2019.

[2] 王鹏程, “代码克隆检测及克隆Bug发现研究,” 中国科学技术大学.

[3] 关军, “软件源代码相似性综合分析系统设计与实现,” 北京邮电大学.

[4] J. R. Cordy and C. K. Roy, “The NiCad Clone Detector,” in *IEEE International Conference on Program Comprehension*, 2011, pp. 219–220.

[5] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V Lopes, “SourcererCC: Scaling Code Clone Detection to Big Code,” 2015.

[6] S. Bazrafshan, “Evolution of Near-Miss Clones,” 2012.

[7] G. Bouma, “Studying the Effects of Code Clone Size on Clone Evolution:,” 2012.

[8] H. Honda, S. Tokui, K. Yokoi, E. Choi, N. Yoshida, and K. Inoue, “CCEvovis: A clone evolution visualization system for software maintenance,” in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, 2019, pp. 122–125.

[9] H. Murakami, Y. Higo, and S. Kusumoto, “Clonepacker: A tool for clone set visualization,” in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2015, pp. 474–478.

[10] S. Baars and A. Oprescu, “Towards automated refactoring of code clones in object-oriented programming languages,” 2019.

[11] A. Alwaqfi, “A Refactoring Technique for Large Groups of Software Clones,” Concordia University, 2017.

[12] D. Rattan, R. Bhatia, and M. Singh, *Software clone detection: A systematic review*, vol. 55, no. 7. Elsevier B.V., 2013.

[13] J. R. Pate, R. Tairas, and N. A. Kraft, “Clone evolution: a systematic review,” *J. Softw. Evol. Process*, vol. 25, no. 3, pp. 261–283, 2013, doi: 10.1002/smr.579.

[14] M. Hammad, H. A. Basit, S. Jarzabek, and R. Koschke, “A systematic mapping study of clone visualization,” *Comput. Sci. Rev.*, vol. 37, p. 100266, 2020.

[15] Q. U. Ain, W. H. Butt, M. W. Anwar, F. Azam, and B. Maqbool, “Recent Advancements in Code Clone Detection--Techniques and Tools,” *IEEE Access*, vol. 7, 2019.

[16] M. Auch, M. Weber, P. Mandl, and C. Wolff, “Similarity-based analyses on software applications: A systematic

literature review," *J. Syst. Softw.*, p. 110669, 2020.

[17] D. Rattan, R. Bhatia, and M. Singh, "Software clone detection: A systematic review," *Inf. Softw. Technol.*, vol. 55, no. 7, pp. 1165–1199, 2013, doi: https://doi.org/10.1016/j.infsof.2013.01.008.

[18] M. F. Zibran and C. K. Roy, "The road to software clone management: A survey," *Dept. Comput. Sci., Univ. Saskatchewan, Saskatoon, SK, Tech. Rep*, vol. 3, 2012.

[19] S. Bharti and H. Singh, "Proactively managing clones inside an IDE: a systematic literature review," *Int. J. Comput. Appl.*, pp. 1–20, 2020.

[20] T. Shippey, D. Bowes, B. Chrisianson, and T. Hall, "A mapping study of software code cloning," 2012.

[21] 苏小红 and 张凡龙, "面向管理的克隆代码研究综述," 计算机学报, vol. 041, no. 003, pp. 628–651, 2018.

[22] A.-F. M. Ali and S. Sulaiman, "A systematic literature review of code clone prevention approaches," *Int. J. Softw. Eng. Technol.*, vol. 1, no. 1, 2014.

[23] K. Wang, L. Zhang, and S. Yan, "A study on code clone evolution analysis," in *Proceedings of 2017 IEEE 8th International Conference on Software Engineering and Service Science*, 2017, pp. 340–345.

[24] D. Rattan and J. Kaur, "Systematic Mapping Study of Metrics based Clone Detection Techniques," in *Proceedings of the International Conference on Advances in Information Communication Technology \& Computing*, 2016, pp. 1–7.

[25] R. V Patil, L. V Patil, S. V Shinde, and S. D. Joshi, "Software code cloning detection and future scope development-Latest short review," in *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, 2014, pp. 1–4.

[26] J. Svajlenko and C. K. Roy, "A survey on the evaluation of clone detection performance and benchmarking," *arXiv*, 2020.

[27] M. Chochlov, "State-of-the-Art Report on Clone Detection," 2017.

[28] A. Paiva and E. Figueiredo, "Do Concern Metrics Support Code Clone Detection?"

[29] K. Wahlroos and others, "Software Plagiarism Detection Using N-grams," 2019.

[30] Q. U. Ain, W. H. Butt, M. W. Anwar, F. Azam, and B. Maqbool, "A systematic review on code clone detection," *IEEE Access*, vol. 7, pp. 86121–86144, 2019.

[31] M. Mondal, C. K. Roy, and K. A. Schneider, "A survey on clone refactoring and tracking," *J. Syst. Softw.*, vol. 159, p. 110429, 2020.

[32] D. Chatterji, "Empirical investigation of causes and effects of code clones," University of Alabama Libraries, 2014.

[33] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering." 2007.

[34] E. V. de Paulo Sobrinho, A. De Lucia, and M. de Almeida Maia, "A systematic literature review on bad smells—5 W's: which, when, what, who, where," *IEEE Trans. Softw. Eng.*, 2018.

[35] A. Bandi, B. J. Williams, and E. B. Allen, "Empirical evidence of code decay: A systematic mapping study," in *2013 20th Working Conference on Reverse Engineering (WCRE)*, 2013, pp. 341–350.

[36] M. Zhang, T. Hall, and N. Baddoo, "Code bad smells: a review of current knowledge," *J. Softw. Maint. Evol. Res. Pract.*, vol. 23, no. 3, pp. 179–202, 2011.

[37] S. Jalali and C. Wohlin, "Systematic literature studies: Database searches vs. backward snowballing," in *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2012, pp. 29–38.

[38] W. Hordijk, M. L. Ponisio, and R. Wieringa, "Review of code clone articles," *Univ. Twente, Netherlands, http//eprints. eemcs. utwente. nl/12257/01/TR-CTIT-08-33. pdf, accessed May*, vol. 18, p. 23, 2009.

[39] A. A. B. Baqais and M. Alshayeb, "Automatic software refactoring: a systematic literature review," *Softw. Qual. J.*, vol. 28, no. 2, pp. 459–502, 2020, doi: 10.1007/s11219-019-09477-y.

[40] A.-W. Harzing and others, "Publish or perish. 2007," *Available http//www. harzing. com/pop. htm. Accessed el*, vol. 14, 2014.

[41] A. W. Harzing, "The Publish or Perish Book: Your Guide to Effective and Responsible Citation Analysis," *Int. Rev. Res. Open \& Distance Learn.*, vol. 13, no. 3, pp. 314–315, 2012.

[42] M. Gusenbauer, "Google Scholar to overshadow them all? Comparing the sizes of 12 academic search engines and bibliographic databases," *Scientometrics*, vol. 118, 2019.

[43] E. V. de P. Sobrinho, A. De Lucia, and M. de A. Maia, "A systematic literature review on bad smells 5 W's: which, when, what, who, where," *IEEE Trans. Softw. Eng.*, vol. PP, no. c, p. 1, 2018, doi: 10.1109/TSE.2018.2880977.

[44] D. Tkaczyk, M. Szostek Paweł and Fedoryszak, P. J. Dendek, and Ł. Bolikowski, “CERMINE: automatic extraction of structured metadata from scientific literature,” *Int. J. Doc. Anal. Recognit.*, vol. 18, no. 4, pp. 317–335, 2015.

[45] Cohen and J., “A Coefficient of Agreement for Nominal Scales,” *Educ. \& Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[46] A. J. Viera and J. M. Garrett, “Understanding interobserver agreement: the kappa statistic.,” *Fam. Med.*, vol. 37, no. 5, pp. 360–363, 2005.

[47] D. Spencer, *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.

[48] T. Zimmermann, “Card-sorting: From text to themes,” in *Perspectives on data science for software engineering*, Elsevier, 2016, pp. 137–141.

[49] A. Begel and T. Zimmermann, “Analyze this! 145 questions for data scientists in software engineering,” in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 12–23.

[50] J. L. Campbell, C. Quincy, J. Osserman, and O. K. Pedersen, “Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement,” *Sociol. Methods \& Res.*, vol. 42, no. 3, pp. 294–320, 2013.

[51] S. Bharti and H. Singh, “Proactively managing clones inside an IDE: a systematic literature review,” *Int. J. Comput. Appl.*, no. 10, pp. 1–20, 2020.

[52] A. Monden, D. Nak Ae, T. Kamiya, S. Sato, and K. Matsumoto, “Software Quality Analysis by Code Clones in Industrial Legacy Software,” 2002.

[53] A. Lozano, M. Wermelinger, and B. Nuseibeh, “Evaluating the Harmfulness of Cloning: A Change Based Experiment,” 2007.

[54] R. Koschke, “Frontiers of software clone management,” 2008.

[55] I. Keivanloo, C. Forbes, and J. Rilling, “Similarity search plug-in: Clone detection meets internet-scale code search,” *IEEE Comput. Soc.*, 2012.

[56] K. Tonscheidt, “Leveraging Code Clone Detection for the Incremental Migration of Cloned Product Variants to a Software Product Line: An Explorative Study.”

[57] Hongzhe *et al.*, “CLORIFI: software vulnerability discovery using code clone verification,” *Concurr. Comput. Pract. Exp.*, vol. 28, no. 6, 2015.

[58] Y. Kamei, H. Sato, A. Monden, S. Kawaguchi, and N. Ubayashi, “An Empirical Study of Fault Prediction with Code Clone Metrics,” 2011.

[59] M. Latapy, “Main-memory triangle computations for very large (sparse (power-law)) graphs,” *Theor. Comput. ence*, vol. 407, no. 1–3, pp. 458–473, 2008.

[60] U. Brandes, “A faster algorithm for betweenness centrality*,” *J. Math. Sociol.*, vol. 25, no. 2, pp. 163–177, 2001.

[61] Q. K. Telesford, K. E. Joyce, S. Hayasaka, J. H. Burdette, and P. J. Laurienti, “The ubiquity of small-world networks,” *Brain Connect.*, vol. 1, no. 5, pp. 367–375, 2011.

[62] M. Bastian, S. Heymann, and M. Jacomy, “Gephi: an open source software for exploring and manipulating networks,” in *Proceedings of the International AAAI Conference on Web and Social Media*, 2009, vol. 3, no. 1.

[63] M. D. Humphries, K. Gurney, and T. J. Prescott, “The brainstem reticular formation is a small-world, not scale-free, network,” *Proc. R. Soc. B Biol. Sci.*, vol. 273, no. 1585, pp. 503–511, 2006.

[64] M. D. Humphries and K. Gurney, “Network ‘small-world-ness’: a quantitative method for determining canonical network equivalence,” *PLoS One*, vol. 3, no. 4, p. e0002051, 2008.

[65] H. Sajnani, *Large-scale code clone detection*. University of California, Irvine, 2016.

[66] Y. Wu *et al.*, “SCDetector: software functional clone detection based on semantic tokens analysis,” in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020, pp. 821–833.

[67] W. Hua, Y. Sui, Y. Wan, G. Liu, and G. Xu, “FCCA: Hybrid Code Representation for Functional Clone Detection Using Attention Networks,” *IEEE Trans. Reliab.*, vol. 70, no. 1, pp. 304–318, 2021, doi: 10.1109/TR.2020.3001918.

[68] C. Liu, Z. Lin, J.-G. Lou, L. Wen, and D. Zhang, “Can Neural Clone Detection Generalize to Unseen Functionalities*f*,” in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 617–629, doi: 10.1109/ASE51524.2021.9678907.

[69] G. Li *et al.*, “SAGA: efficient and large-scale detection of near-miss clones with GPU acceleration,” in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2020, pp. 272–283.

[70] T. Nakagawa, Y. Higo, and S. Kusumoto, “NIL: Large-Scale Detection of Large-Variance Clones,” in *Proceedings of the*

29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021, pp. 830–841, doi: 10.1145/3468264.3468564.

[71] P. Wang, J. Svajlenko, Y. Wu, Y. Xu, and C. K. Roy, "CCAligner: A Token Based Large-Gap Clone Detector," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 1066–1077, doi: 10.1145/3180155.3180179.

[72] J. Svajlenko and C. K. Roy, "Fast and Flexible Large-Scale Clone Detection with CloneWorks," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 2017, pp. 27–30, doi: 10.1109/ICSE-C.2017.3.

[73] H. Sajnani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes, "SourcererCC: Scaling code clone detection to big-code," *Proc. - Int. Conf. Softw. Eng.*, vol. 14-22-May-, no. 1, pp. 1157–1168, 2016, doi: 10.1145/2884781.2884877.

[74] P. Wang, J. Svajlenko, Y. Wu, Y. Xu, and C. K. Roy, "CCAligner: A token based large-gap clone detector," *Proc. - Int. Conf. Softw. Eng.*, pp. 1066–1077, 2018, doi: 10.1145/3180155.3180179.

[75] H. Liu, Z. Yang, Y. Jiang, W. Zhao, and J. Sun, "Enabling clone detection for ethereum via smart contract birthmarks," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, 2019, pp. 105–115.

[76] K. W. Nafi, T. S. Kar, B. Roy, C. K. Roy, and K. A. Schneider, "CLCDSA: Cross Language Code Clone Detection using Syntactical Features and API Documentation," *2019 34th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, pp. 1026–1037, 2020, doi: 10.1109/ase.2019.00099.

[77] D. Perez and S. Chiba, "Cross-language clone detection by learning over abstract syntax trees," *IEEE Int. Work. Conf. Min. Softw. Repos.*, vol. 2019-May, pp. 518–528, 2019, doi: 10.1109/MSR.2019.00078.

[78] C. V Lopes *et al.*, "DéjàVu : A Map of Code Duplicates on GitHub PETR MAJ , Czech Technical University , Czech Republic Additional Key Words and Phrases : Clone Detection , Source Code Analysis ACM Reference Format : Forge have transformed how source code is shared . Creating," *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, 2017.

[79] M. Mondal, C. K. Roy, and K. A. Schneider, "Identifying Code Clones Having High Possibilities of Containing Bugs," *IEEE Int. Conf. Progr. Compr.*, pp. 99–109, 2017, doi: 10.1109/ICPC.2017.31.

[80] R. Yue, Z. Gao, N. Meng, Y. Xiong, X. Wang, and J. D. Morgenthaler, "Automatic Clone Recommendation for Refactoring Based on the Present and the Past," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 115–126, doi: 10.1109/ICSME.2018.00021.

[81] C. K. Roy, M. F. Zibran, and R. Koschke, "The vision of software clone management: Past, present, and future (Keynote paper)," in *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, 2014, pp. 18–33, doi: 10.1109/CSMR-WCRE.2014.6747168.

张迅晖(1993—),2017年获得国防科技大学软件工程硕士学位,现为国防科技大学软件工程在读博士研究生,主要研究方向为软件仓库挖掘,开发者贡献度量,代码克隆.

钟岩(1998—),2020 年获得华中科技大学学士学位,现于国防科技大学攻读硕士学位,主要研究方向为软件仓库挖掘,代码克隆.

王涛(1984—),2014 年获得国防科技大学计算机科学博士学位,现为国防科技大学副教授,主要研究方向为软件仓库挖掘,基于群体智能的软件工程,开源软件生态.

张晏芝(1997—),2019年获得云南大学信息安全学士学位,现为国防科技大学在读硕士研究生,主要研究方向为智能化软件工程,代码克隆检测.

余跃(1988—),2016 年获得国防科技大学软件工程博士学位,现为国防科技大学副教授,主要研究方向为实证软件工程,群体化开发,开源软件生态

王怀民(1962—),1992年获得国防科技大学软件工程博士学位,现为国防科技大学教授,中国科学院院士,主要研究方向为可信计算,群体智能,分布式计算.