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

面向开源生态的软件数据挖掘技术研究综述

尹 刚[1], 王 涛[1], 刘冰珣[1], 周明辉[2], 余 跃[1], 李志星[1], 欧阳建权[3], 王怀民[1]

[1](国防科学技术大学 计算机学院, 长沙 中国 410073)

[2](北京大学 信息技术学院, 北京 中国 100871)

[3](湘潭大学 信息工程学院, 湘潭 中国 411105)

通讯作者: 尹刚, E-mail: yingang@nudt.edu.cn

摘 要: 全球开源软件生态中孕育的大众化软件生产模式正快速形成一种新型的软件生产力，在软件开发和应用各个环节发挥了巨大作用。大众化软件生产的数据规模日趋庞大、协同范围不断扩展、管理模式高度精简，这些全球化特征使其在软件复用、协同开发、知识管理等环节面临诸多挑战，迫切需要新的理论指导和工具支持。本文首先界定了大众化软件生产活动的分布范围、基本过程和数据形态，然后从软件复用、协同开发、知识管理三个核心环节对开源社区数据挖掘技术的研究工作进行了归类与分析，最后总结了该领域研究工作存在的问题和未来发展趋势。

关键词: 开源社区；软件仓库；开源软件；数据挖掘

中图法分类号: TP311

中文引用格式:

英文引用格式:

Survey of Software Data Mining for Open Source Ecosystem

Yin Gang[1], Wang Tao[1], Liu Bing-Xun[1], Zhou Ming-Hui[2], Yu Yue[1], Li Zhi-xing[1], Ouyang Jian-Quan[3], Wang Huai-Min[1]

[1](School of Computer, National University of Defense Technology, Changsha 410073, China)

[2](School of Information Technologies, Peking University, Beijing 100871, China)

[3](School of Information Engineering Technologies, Xiangtan University, Xiangtan 411105, China)

Abstract: Crowd-based software production model in global open source software ecosystem is rapidly becoming a kind of new software productivity, and has great impacts on many stages of software development and applications. Crowd-based software production generates large amounts of software data, continuously expands its collaboration scopes, and highly simplifies its project management. These globalization features make crowd-based software production face many challenges in software reuse, collaboration development, and knowledge management, which urgently require new theories and tools to support. In this paper, we firstly classify the distribution, basic process and data form of crowd-based software production activities. Then we classify and analyze the studies of software communities data mining technology from the three core aspects—software reuse, collaborative development and knowledge management. Finally, we summarize the problems and future trends of research works in this field.

Key words: open source communities; software repositories; open source software; data mining

基金项目: 国家重点研发计划(2016YFB1000805); 国家自然科学基金(61472430)

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

1 引言

近年来，开源软件对软件开发技术、运行形态和服务模式都产生了前所未有的影响，软件技术已进入网络时代[1][2]。截至 2017 年 9 月，社交编程网站 Github(github.com)托管了超过 6000 万个软件版本库，约 2500 万用户参与到开源项目的开发和贡献，有效支持了 Linux、Rails、jQuery 等知名开源软件对全球数百万自由开发者贡献的汇聚；近年来发展迅速的软件问答社区 StackOverflow(stackoverflow.com)已发布了约 1500 万个软件问题，每月有约 5000 万名的活跃用户，如此庞大的用户群形成了强大的问题解答能力，平均响应时间仅为 11 分钟；在国内，手机操作系统 MIUI 在其用户社区的数百万发烧友帮助下保持了每周更新一次的惊人迭代速度，这是小米手机短期内成功爆发的关键因素。开源软件已经形成了一种全球化创新发展的生态系统。其中孕育的大众参与的软件开发模式正逐步形成一种基于互联网的新型软件生产力，并已在软件开发和应用的各个环节发挥了巨大作用。

这种大众化软件生产活动以开源社区为平台，其生产过程和制品数据对外开放，允许不同类型的生产者参与其中，能够有效汇聚来自互联网的创意和贡献。以开源软件为例，分布在世界各个角落的软件开发者和用户在少数核心人员的协调下，利用业余时间打磨出诸如 Linux、Apache 和 Firefox 等众多商业级软件产品。同传统工业化软件生产相比，大众化软件生产具有以下特征：1) 开发数据和应用数据高度开放且规模巨大。目前支持大众化软件生产和应用活动的开源社区包含了大量有价值的数据，如软件代码[3]、软件版本[4]等软件制品和过程数据，以及软件问答[5]、软件评价[6]等软件交流和反馈数据，这些数据规模巨大，且具有高度开放性；2) 协同范围不断扩展。大众化生产的软件项目的主要推动力来自项目外部开发者和用户的贡献(如代码贡献[7]、缺陷报告[8]等)，尤其是随着 Pull-Request 等新型开发机制[错误!未找到引用源。]的不断发展，外部贡献的比率不断扩大，软件生产活动将越来越依赖于项目的外围大众；3) 项目管理模式高度精简。大众化软件生产通常没有规范的开发计划、过程说明，甚至没有设计文档[7]，很多成功项目往往依赖于个别天才程序员[10]。大众化软件生产的上述特征一方面有助于释放大众创意、汇聚大众贡献，另一方面，由于生产力的快速提升，软件复用、协同开发、知识管理等环节的现有生产模式开始面临新的挑战：

(1) 可复用软件资源快速增长且广泛分布，软件资源的利用从对封闭资源集合的检索转向对大规模开放资源的可信定位。开源社区中存在大量可复用资源，包括 API[11]和代码片段[12]等。这些资源规模巨大、高度分散、良莠不齐，难以用传统方式进行管理和评估[13][14]，软件的可信性需求日益迫切[2]。如何在开源社区数据中有效定位高度相关且高可信的可信软件资源，成为软件复用环节迫切需要解决的瓶颈问题。

(2) 外部贡献比例不断增大，软件开发任务的分配从面向团队的内部指派转变为面向开放群体的外部贡献。随着大众参与模式的不断改进，来自外部开发者和用户的缺陷报告[8]、合并申请[错误!未找到引用源。]、功能建议[15]等外部贡献不断增多，其中出现的“陌生参与者”、低质量外部贡献等问题使现有开发任务指派技术[16][17]面临新的挑战，如何利用不同开源社区中的信息预测大众参与的任务优先级和潜在参与者成为软件协同开发能否高效推进的关键。

(3) 开源社区形成了海量软件知识文本，软件开发过程的知识管理从技术文档的人工维护转向基于社区的知识推荐。开源社区中蕴含着大量关于软件编程[18]、API 使用[19]、用户需求[20]、软件缺陷[21]的相关知识，这使软件需求文档和缺陷报告等的自动维护成为可能，对汇聚大众创意、精简项目管理具有重要意义。但是，如何从社区数据中获取高质量软件开发相关知识并推荐给软件项目，是实现软件知识管理的新挑战。

大众化软件生产面临上述新挑战和新问题具有一定的必然性，其产生的根本原因在于大众化软件生产本质上是一种全球化软件工程活动，需要从更大的范围去认识和把握其内涵和外延。

近年来国内外学者基于开源软件生态数据和相关软件过程数据，围绕上述问题开展了大量软件数据挖掘的研究工作。其中，软件资源挖掘方向的研究关注可信评估和资源定位；开发数据挖掘以最佳实践分析、缺陷优先级分析和技术专家识别等为核心；软件知识推荐则关注问答挖掘和 API 示例挖掘等软件知识挖掘技术。

本文第 2 节对大众化软件生产的活动范围、基本过程和数据类型进行概述；第 3 节提出开源社区数据挖掘的基本框架，并依据该框架从软件复用、协同开发和知识管理三个方面对开源社区数据挖掘的研究工作进

行综述；第 4 节对该领域当前研究存在的问题和机遇进行总结，对未来工作进行展望；第 5 节总结全文。

2 大众化软件生产

2.1 开源社区

随着开源生态系统的发展和演变，开源社区的概念内涵也不断拓展。目前，开源社区主要可以划分为开源协同开发社区和开源知识分享社区，两类社区中的数据从不同阶段、不同侧面反映了软件开发和应用等活动的历史数据和最新动态。如前所述，这两类社区中都包含大量有价值的数据，其中分布着大量记录了软件开发、分享、应用、维护等重要信息的不同维度的数据实体，本文称之为**软件信息实体，简称信息实体或实体**。其中，软件协同开发社区(如 SourceForge、Github 等)中的信息实体主要为开发过程制品和软件资源，如软件版本[3]、缺陷报告[8]、提交记录[22]等，这些信息实体之间具有开发活动形成的相对稳定的结构关系，如 API 调用关系[23]、开发者协同关系[24]等；软件知识分享社区(如 StackOverflow、OsChina、CSDN 等)则包含了软件的文本描述[25]、社会化标签[26]、用户评论[6]、软件问答[5]等，这些实体往往是对协同开发社区中信息实体的应用和评价，具有更为丰富的语义关系。

这两类社区不仅通过软件名称、API、标签、描述等信息广泛连接，而且通过大量同时活跃于这两类社区的参与者产生互动[5]。这些实体及其关系为深入理解大众化软件生产模式的核心机理、突破传统软件开发的效率与质量瓶颈提供了有效的解决思路。例如，软件知识分享社区中存在大量能够反映软件可信属性的数据[6]，对可信资源的定位具有重要价值；大众参与者在不同社区的行为和贡献可用于解决“陌生参与者”问题；而大众参与行为涌现出来的知识及其活跃度则为任务优先级预测和自动化知识管理提供了新的视角和途径。可见，互联网中的软件协同开发社区和知识分享社区中的信息实体具有很强的互补性和关联性，对其综合分析和利用将有助于解决可信资源定位、开发任务预测、社区知识推荐等大众化软件生产面临的新问题。软件在开发过程中产生和记录的软件演化和活动者组成了软件工程的大数据[7][8]。可以说，开源社区已经呈现了大数据特征，对开源社区数据的研究其实质是一种大数据条件下的工程技术研究，具有十分重要的研究意义。

2.2 大众化生产的关键机制

大众化软件生产是以核心团队为中心、外部贡献为主要推动力开展的，其关键机制在于如何高效地组织软件协同开发活动、合理地分配开发任务，以及充分利用软件知识或复用软件代码。

在大众化生产过程中，因为参与人员自由度高、协作程度松散，外部贡献的代码参差不齐，需要花费大量的时间和精力验证，这已然成为制约软件生产和进行有效协同开发活动的关键因素，所以挖掘开发任务的优先级和为开发任务推荐合适的开发者（即开发任务预测）成为要关注的重点内容；同时，存在于软件知识分享社区内的软件知识虽然为软件开发文档提供了丰富的来源，但是如何从大量的知识中提取真正需要的内容是现阶段要解决的问题；除此之外，协同开发社区内包含海量的软件项目、软件代码等不同粒度的可复用软件资源，由于大众软件开发活动缺乏严格的管理，因而在这些资源中定位出高可信的相关软件资源成为大众化软件开发的需求。

因此，围绕大众化软件生产面临的挑战和问题，相关研究从多个角度开展了大量研究。本文重点关注其在软件资源复用、开发任务预测和软件知识推荐这三个方面的研究进展。

2.3 开源软件社区的数据特点

软件协同开发社区指的是以 SourceForge、GitHub 等为代表的管理软件开发、寻找开源代码、支持软件项目托管和大众软件开发的在线平台。以 SourceForge 为例，该社区包含 50 万个软件项目，参与者多达数百万人。可以说软件协同开发已经是软件开发的一种主流趋势，而软件协同开发社区则是实现这个过程的主要平台，在软件协同开发中有举足轻重的地位。以 SourceForge 为例，一个在 SourceForge 上托管的软件工程项

目可以在这个社区里看到项目相关的文本文档、所依赖的编译及运行环境、开发使用的工具包、开发者的邮件列表、用户使用的评价和讨论以及项目的代码版本库等信息。大众贡献者和项目管理者依托协同开发社区开展软件生产活动，例如外围贡献者可以根据软件的描述找到自己感兴趣的软件项目，结合项目的缺陷列表，完成软件缺陷修复相关的开发工作；项目的管理者会判断贡献者提交内容的质量和价值，进而规划整个项目的发展方向。软件协同开发数据即从这样软件协同开发社区里而来，它包括了协同开发活动过程中的代码版本库、缺陷库、提交信息、邮件列表、配置文件等软件仓库数据[06][11]。此类协同开发数据蕴含了软件开发的历史信息和未来动态，通过对数据的挖掘与分析能帮助用户有效的进行协同开发，提升用户的专业技能。例如，在 GitHub 里，用户可以围绕评论，代码质量，社区里程碑，人物关系等信息管理协同开发，学习知识以及获取声誉。这种大型的协同开发社区数据能够有效地支持改革创新，共享知识和建立社区的多个途径。

随着 StackOverflow，OsChina，CSDN 等这样软件知识分享网站的出现，它们为用户提供了比传统交流方式更多的社交刺激，比如通过丰富的网页环境来协同存储和管理内容，以及一个能向同行或者潜在的项目参与者更生动展示他们的知识和专业的地方。可以看出，软件知识分享社区为在线用户提供了一条快速获取专业技能的途径和一个丰富的交流信息平台，对开发人员非常有益。以 StackOverflow 为例，它主要是以提问和回答的形式来实现用户之间大部分交流。问题可以通过添加标签来进行类型分类，而回答则是根据参与者对答案的赞同数量来进行排名。答案获得的赞同数量越多，其参考价值也就越大。软件知识分享社区里的数据包括软件资源的文本描述，软件问答，软件评论，社会化标签以及相关帖子等等。这些数据极大的帮助了软件开发。StackOverflow 的问答在代码评论和概念性的问题等方面非常有效，也可以帮助开发者找到合适的代码示例以及 API 使用示例等有用信息，从而帮助个人和企业利用这些信息进行相关开发。

这两类社区虽然独立运行，但是两类社区数据并不是独立存在的，而是紧密相联的。很多时候，GitHub 的开发者在 StackOverflow 上寻求解决他们的技术挑战的方案；相似的，他们可以参与 StackOverflow 回答其他人提出的问题。有研究表明，同一用户在 GitHub 和 StackOverflow 上的活动存在着这样的关系：积极的 GitHub 提交者往往在 StackOverflow 上极少问问题，但提供了比一般用户更多的答案；StackOverflow 上的活动比率与 GitHub 上代码的变化活动是相关的。由此不难看出，协同开发数据和知识共享数据有很强的互补性和联系性，结合它们自身的特点，综合分析利用，可以解决可信资源定位、开发任务预测、社区知识推荐等大众化软件生产面临的新问题。

表 1 开源协同开发社区数据分类

数据名称	描述	典型数据源
源代码	位于软件版本库，包含软件不同版本的源代码	版本控制系统,如 CVS, SVN, Git 等
提交日志	位于软件版本库,记录了源代码的变更历史	版本控制系统,如 CVS, SVN, Git 等
缺陷库	软件问题的发布和解决情况的记录	缺陷/问题跟踪系统,如 Bugzilla, Trac, Jira 等
邮件列表	开发者和用户间的面向特定主题的邮件内容	某些站点提供的邮件列表存档，如 lkml.org 等
特征申请	用户对软件功能的需求和创意反馈	某些开源社区提供,如 SourceForge 的 Feature Request 工具
合并申请	开发者贡献代码的合并申请及相关记录	某些开发社区提供,如 GitHub 的 Pull-Request 工具

表 2 开源知识分享社区数据分类

名称	描述	典型数据源
软件问答	关于软件编程、使用等的问题和回答	软件编程问答社区，如 StackOverflow，CSDN 等
软件标签	标识软件技术和功能特征的标签(Tag)	某些社区帖子提供的标签数据，如 OSChina 等
社交互动	面向社交网络的开发者社交数据	一些网站提供的社交平台和工具，如 follow 和@等数据信息
特征列表	软件的功能和列表	开发者对软件的描述信息，如 CSDN，51CTO 等
其他文本	博客、新闻等关于软件的动态和评述	一些软件相关的新闻门户网站，如 CNBlog，Slashdot 等

3 开源社区数据挖掘技术

近年来，软件资源库挖掘(mining of software repositories)成为当前国际软件工程领域最为活跃的研究方向之一，其中出现了大量围绕资源可信评估和定位、开发任务分配和社区知识管理等问题开展的数据挖掘研究，

表 3 从软件资源挖掘、软件开发任务挖掘和软件知识挖掘三个方面对这些研究工作进行了总结和梳理，并根据其采用的数据挖掘和分析技术进行了分类，较为直观地展示了这些研究所涉及的数据类型和方法技术。为了增加参考价值和覆盖面，表中也增加了企业内部软件项目数据挖掘的研究[46][55][61][63][64]。

3.1 软件资源挖掘技术

软件资源库挖掘的主要任务是帮助开发者找到高质量的可复用软件资源。近年来，以开源软件为主体的互联网软件资源规模迅速膨胀，为开发人员复用开发提供了坚实基础。但是，如何从海量互联网资源中快速准确定位到可信的资源是开源软件复用面临的挑战性问题。当前研究主要从开源软件可信评估和软件资源定位两个方面展开研究。

3.1.1 开源软件可信评估

工业化软件质量评估模型[14]主要关注产品本身及其严格定义的软件开发过程，但是开源软件项目管理高度简化的特点使得传统方法难以有效运用。针对开源软件质量评估问题，早期一些学者提出了 OpenBRR 模型[28]、QSOS 模型[29]和 SQO-OSS 模型[30]等，将软件代码本身和软件项目社区等结合起来，定义了包括可用性、安全性、可扩展性、项目文档等一系列复杂的指标对开源软件质量进行度量。这些评估模型虽然全面涵盖了开源软件从开发过程到开发社区等各个方面，但是评估指标过于复杂，整个评估过程难以实现自动化。为此，Lavazza[31]和 Bauer[32]等人分别从代码编写风格和代码行数、最长函数行数等指标对软件代码进行静态分析，从而对软件质量进行评估。然而这些方法仅关注软件的内部质量，无法对软件的可用性和性能等方面进行度量。Zou 等人[6]则从软件用户的角度入手提出了一种基于互联网用户评论的软件质量度量方法，对互联网上海量用户评论的正面和负面倾向进行情感分析，并基于评论数据从轻量性、易用性、稳定性等六个方面对软件质量形成综合评估，该研究对本文基于社区信息的软件可信资源定位研究具有重要借鉴意义。

3.1.2 软件资源定位

在软件资源定位方面，研究人员从不同层面、不同角度提出了帮助开发者快速定位所需资源的技术，包括软件自动分类、软件特征定位以及软件可追踪性分析技术。

基于分类的软件资源定位以软件类别为粒度，将具有相似功能的软件聚合在一起，能够有效提高软件的查找效率。当前相关研究的基本思想是给定一个软件分类体系，基于软件代码或描述等数据，利用支持向量机(SVM)[33][34]、潜在语义索引(LSI)[35]、概率主题模型(LDA)[36]等方法对软件进行分类。这些方法主要针对具体软件的分类，在面向互联网规模的软件定位时面临可扩展性差、分类效率低等性能问题。为此，Xia[26]、Wang[37]等人基于软件在线标签及描述数据，通过标签推荐和分类实现大规模软件资源的组织。软件分类研究主要面向软件资源的组织管理，分类过程中聚焦于软件本身的代码或描述，而软件的质量、用户评论等来自其他社区的可信数据并未被作为分类属性的一部分加以考虑,因此分类结果仅仅完成了相似功能软件的聚合，而对无法对同一类别下的软件根据可信进行排序。

基于特征的软件资源定位是以特征为粒度的软件资源定位，具体包括特征分析和特征定位两个步骤。传统的特征分析技术主要从软件源代码、注释等数据中提取软件的领域特征[38][39]，Dumitru[25]和 Davril[40]等人则首次提出基于互联网社区文本数据的特征分析方法，利用 Softpedia.com 和 CNET.com 中海量软件的在线特征描述信息构建软件的特征模型。在此基础上，特征定位的难点在于自然语言的特征描述与编程语言的代码实现之间存在鸿沟。为此，人们先后采用了文本分析、静态代码分析、动态运行轨迹等多种方法实现特征定位[41]，并提出了一系列组合方法[42][43][44][45]。其中具有代表性的是 Dit 等人[45]设计了一个面向特征定位的数据融合模型，将代码文本分析、动态执行轨迹以及代码依赖关系结合起来，利用链接分析技术(HITS 或 PageRank)对挖掘结果进行优化。当前基于特征的软件资源定位主要针对具体软件，从软件代码中定位到特定特征的实现。但是在互联网环境下，实现相同特征的软件资源通常数量巨大，当前研究仅从特征描述与特征实现之间的相关性强度进行排序，而并未把特征实现代码的质量、易用性等可信属性纳入考量。

表 3 代表性工作的比较与分类

分类	任务	文献	开源社区数据								方法与技术												
											信息检索		机器学习			主题模型		NLP		API	社交	其
			代码	缺陷	邮件	问答	特征	日志	社交	标签	JS	VSM	朴素贝叶斯	KNN	SVM	LDA	LSI	文本	代码	调用	网络	他
软件资源挖掘	特征分析	[25]		·	·	·	●	·	·	·	·	·	·	●	·	·	·	·	·	·	·	·
		[38]	●	·	·	·	●	·	·	·	·	·	·	·	·	·	·	·	●	·	·	·
		[39]	●	·	·	·	●	·	·	·	·	·	·	·	·	●	·	·	·	·	·	●
		[40]	·	·	·	·	●	·	·	·	·	·	·	·	·	·	·	●	·	·	·	●
	特征定位	[42]	●	●	·	·	●	·	·	·	·	●	·	·	·	·	·	●	·	·	·	·
		[43]	●	●	·	·	●	·	·		·	·	·	·	·	·	●	●	·	·	·	·
		[44]	●	·	·	·	●	·	·	·	·	·	·	·	·	●	·	·	·	·	·	·
		[45]	●	●	·	·	●	·	·	·	·	·	·	·	·	·	●	·	●	·	·	·
	可追踪性分析	[46]	●	●	·	·	·	·	·	·	·	·	·	·	·	●	●	·	·	·	·	·
		[47]	●	·	●	·	·	·	·	·	·	●	·	·	·	·	●	●	·	·	·	·
		[48]	●	●	●	·	●	·	·	·	●	●	·	·	·	·	·	·	·	·	·	·
		[49]	●	·	·	·	●	·	·	·	●	●	·	·	·	●	●	·	·	●	·	●
		[50]	●	·	·	·	●	·	·	·	●	●	·	·	·	·	●	·	·	·	·	·
软件任务挖掘	历史缺陷预测	[53]	·	●	·	·	·	·	·		·	·	●	·	·	·	·	●	·	·	·	·
		[54]	·	●	·	·	·	·	·		·	·	●	●	●	·	·	●	·	·	·	·
		[55]	·	●	·	·		·	·	·	·	·	·	·	·	·	·	·	·	·	·	●
		[56]	·	●	·	·	·	●	·	·	·	·	·	·	·	·	·	·	·	·	·	●
		[57]	·	●	·	·	·	●	·	·	·	·	·	·	●	·	·	·	·	·	●	·
	缺陷指派	[17]	·	●	·	·	·	·	·	·	·	·	·	·	●	·	·	·	·	·	·	●
		[58]	·	●	·	·	·	·	·	·	·	·	·	·	●	●	·	·	·	·	·	·
		[59]	·	●	·	·	·	·	·	·	·	·	·	●	·	●	·	·	·	·	·	·
		[60]	·	●	·	·	●		●	·	·	·	·	·	·	·	·	·	·	·	●	·
		[61]	●	●	·	·	·	·	·	·	·	●	●	●	●	·	·	·	·	·	·	●
		[62]	●	●	·	·	·	·	·	·	·	·	·	·	·	·	●	·	·	·	·	·
	专家识别推荐	[24]	·	·	·	·	·	·	●		·	·	·	·	·	·	·	·	·	·	·	●
		[63]	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●
		[64]	·	●	·	·	·	·	·	·	·	·	·	·	·	●	·	·	·	·	·	·
		[65]	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●
		[66]	·	·	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●
		[67]	·	●	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●	·
软件知识挖掘	问答挖掘	[18]	·	·	·	●	·	·	·	●	·	●	·	·	·	·	·	·	·	·	·	·
		[69]	●	·	·	●	·	·	·	·	·	·	·	·	·	●	·	·	·	·	·	·
		[70]	●	·	·	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●
		[71]	·	·	●	●	·	·	·	·	·	·	·	·	·	·	●	●	·	·	·	·
		[72]	●	·	·	●	·	·	·	·	·	·	·	·	·	·	·	●	·	·	·	●
	API示例	[19]	●	·	·	·	·	·	·	●	·	·	●	·	·	·	·	·	·	·	·	·
		[73]	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●	·	●
		[75]	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●	●	·	·
		[76]	●	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	●	·	●

基于链接的软件资源定位主要是对不同类型资源和制品的可追踪性分析，定位与其相关的软件生产活动产生的缺陷报告、说明文档、交流讨论等资源。目前，多数可追踪性恢复研究采用文本挖掘方法[46][47][48]，但是文本挖掘方法很大程度上依赖于文本信息质量以及查询输入质量。为此，Dasgupta[49]和 Capobianco[50]等人分别将语义关联和词性分析纳入进来，通过扩展文档库和仅保留名词的方法，有效提高了文本挖掘方法的效果。不同于上述文本挖掘方法，Begel 等人[51]通过对项目开发活动进行分析建立人和制品的关联网络，并设计了一种基于路径的查询语言，能够用于相关人员、代码、缺陷等的发现和定位。

综上所述，当前软件资源可信定位相关工作从可信评估和资源定位两个角度独立开展研究，取得了大量成果，对本文具有重要借鉴意义。但是，在互联网环境下针对大规模软件资源，传统可信度量方法过于复杂，需要较大代价。同时，资源定位方法主要针对定位的准确性，缺乏在定位过程中考虑软件质量和用户满意度等重要可信属性。如何将资源可信分析与定位结合起来，权衡资源的可信性和定位的准确性，实现大规模软件资源的可信定位，是该方向有待深入研究的新问题。

3.2 软件开发任务挖掘技术

软件开发任务挖掘主要是通过数据挖掘的手段对软件项目管理提供辅助决策，以降低开发活动的管理开销。在大众化软件生产场景下，来自外部开发者和用户的贡献不断增多，如何有效地管理这些贡献的问题尤其突出。例如，截至 2017 年 10 月 Mozilla 一共收到 140 万个缺陷报告，平均每天收到 300 多个[52]。这就迫切需要对这些报告进行甄别和处理以实现快速响应。近年来该方向的研究点包括软件缺陷优先级分析、开发任务指派和技术专家识别等。

3.2.1 缺陷优先级预测

缺陷优先级预测能够帮助项目管理者按照缺陷严重程度对涌入的缺陷报告进行排序处理，能够加快对严重缺陷的处理。为分析缺陷报告的质量，Hooimeijer 等人[16]对 27000 个缺陷的外部特征进行统计分析并设计了缺陷报告质量描述模型，以预测某缺陷报告是否能在给定时间内被指派，该研究对缺陷报告系统以及缺陷提交者所应该强调的特性具有重要启示。

基于历史缺陷的优先级信息进行优先级预测是普遍使用的方法，Lamkanfi[53][54]利用历史缺陷报告的描述及严重性信息构建分类器自动判断缺陷报告是否严重，Yu 等人[55]则基于缺陷报告中的类别、里程碑等信息采用神经网络模型对缺陷优先级的预测。不同于直接以优先级为类别构建分类器的优先级预测方法，Kim 等人[56]将缺陷类别作为预测缺陷优先级的指标，首先统计分析不同类别报警信息的优先级，然后对新报警进行分类并根据类别确定其优先级。Zanetti 等人[57]则将缺陷提交者作为预测缺陷报告有效性的重要指标，根据“指派”和“抄送”关系构建提交者网络，使用网络中心度等 9 个测度对缺陷提交者进行度量，并据此预测缺陷的有效性。

现有方法主要根据缺陷的严重性(severity)对其进行分级，严重性越高的缺陷其优先级就越高。其出发点是软件质量，而不是用户需求及其活跃度。但是很多影响用户体验的缺陷虽然不是严重错误但可能会使用户失去兴趣，最终导致项目发展受到影响。如何融合多个社区的数据，综合考量缺陷本身的严重程度和用户需求对缺陷优先级进行分析是值得研究探讨的重要方向。

3.2.2 自动化缺陷指派

自动化缺陷指派机制能够有效减轻大众化软件生产中项目管理者的负担，提高任务完成的效率和质量。基于分类的缺陷指派技术利用了历史缺陷处理轨迹或者开发者的开发活动信息，对新产生的缺陷进行开发者指派[17][58][59][60][61]。其中，Naguib 等人[58]综合利用缺陷的评阅、指派、解决等信息刻画开发者的技术能力、项目角色及参与程度，从而提高推荐的准确性。Xuan 等人[60]则在文本分类方法基础上，将网络分析方法结合起来，基于开发者在历史缺陷中的同现关系构建开发者关联网络，并利用该网络中各开发者的网络位置对分类挖掘结果进行优化。基于定位的缺陷指派回避了上述工作中复杂的机器学习训练过程，通过分析引发缺陷的源代码文件来寻找合适的缺陷修复者。Vásquez 等人[62]针对变更请求的指派，首先定位与之相关的代码文件及所有者，然后根据所有者在相关文件中出现的频次对其进行排序和推荐。相关研究对缺陷指派局限在

项目内部开发者，开发者的技术能力是最重要的考量指标。在大众生产模式下，外部贡献越来越多，如何充分发掘外部贡献者使其参与缺陷解决对于推动项目发展具有重要意义。同时，除了技术能力外，开发者的活跃程度或者参与意愿[62]也会影响缺陷解决的质量和效率。因此，在做缺陷指派时，应将其作为重要因素纳入考量。

3.2.3 领域专家识别与合作者推荐

事实上，开发任务分配除了技术上的匹配外，开发者的能力也成为越来越重要的因素。准确的专家识别对高度分散化、全球化的软件生产活动的管理具有重要意义。Mockus 等人于 2002 年首先提出了基于数据挖掘的软件专家定位技术[63]。近两年基于开源社区数据的软件专家推荐成为研究热点，相关研究工作基于不同测度对开发者的技术能力进行度量。Nguyen 和 Ma 等人[64][65]分别从实现能力(Implement Expertise)和使用经验(Usage Expertise)两个角度，通过分析开发者解决缺陷的时间分布和对函数的使用频率来度量开发者的技术水平。为推荐合适的合作者和指导者，Surian 等人[24]基于 SourceForge 构建“开发者-项目-类别/编程语言”网络，然后通过在项目、类别/编程语言中的同现来定义开发者之间的社交距离，进而为给定开发者推荐合适的合作者。Canfora 等人[66]利用邮件列表和版本库数据，从用户的交互活跃度和技术活跃度两个角度入手进行分析，为项目新成员推荐合适的指导者。Gharehyazie 等[67]则利用开发者在项目中的社交网络分析其成为某一个项目贡献者的可能性。

当前领域专家识别研究主要从开发者的项目参与情况进行分析，忽略了开发者在知识分享社区中的活动，无法准确度量那些刚加入某项目的开发者的能力(即“陌生参与者”问题)。如何将开发者在不同社区中的活动整合起来，全面度量开发者能力是值得研究探讨的重要问题。

综上所述，当前开发任务挖掘的研究基于缺陷历史数据对优先级进行预测，并利用开发者的历史活动信息从开发能力、使用经验、沟通交互能力等不同的测度提出了度量开发者技术能力并进行缺陷指派的方法，这些工作为本文的研究提供了借鉴。但是，当前研究主要聚焦于一个项目或者社区内部，对那些尚未积累足够数据的信息实体难以准确分析和度量。在大众化软件生产模式下，与信息实体关联的数据往往散布在互联网不同社区中而不是局限在一处。例如，与缺陷相关的数据不仅仅存在于项目缺陷管理工具中，也散布在其他类似 StackOverflow 等社区的用户讨论评价中；同一个开发者不仅活跃在协同开发社区中参与项目开发，同时也活跃在知识分享社区中贡献开发知识和经验[5]。这些数据聚合在一起才形成了对信息实体的完整描述。针对具体问题，如何将分散的相互关联的社区数据有机整合与利用，从而进行完整全面的度量是非常有意义的研究工作。

3.3 软件知识挖掘技术

软件开发的一项重要任务是开发过程的知识管理[68]，虽然软件企业以知识管理作为维护其核心资产的手段，在互联网环境下其对改善软件项目的易维护性和大众参与度具有重要意义。近期的一些相关工作对开源社区的问答知识和 API 示例等数据进行了挖掘，另有一些工作研究了如何把软件知识与开发环境实现有效集成。

3.3.1 问答知识挖掘

随着以 StackOverflow 为代表的知识分享社区的繁荣，开源软件知识管理的内涵和范围都发生了变化。这些社区日益成为开发者获取知识的重要来源。Allamanis 等人[69]对 StackOverflow 中提问的类型、概念、主题等分析发现该社区中的问题涵盖了不同编程语言、开发平台且涉及到各种类型的问题。Parnin 等人[70]分析发现 StackOverflow 中关于 Android API 的讨论涵盖了 Android 中 87%的类，而且这些讨论被用户浏览了超过 7000 万次。随着这些社区的迅速发展，研究人员开始关注以开源社区为单位的软件开发知识的抽取和挖掘，以支持知识的高效获取、关联和推荐等，包括相关问答的检索[18]、常见问题解答的自动抽取[71]、代码注释自动生成[72]、API 关联文档获取[19][73]、知识分享社区与开发环境的集成[74]等典型研究。

3.3.2 API 示例挖掘

如何使用某个 API 是开发者经常面对的问题，为快速准确检索 API 使用示例，Xie 等人[75]将代码结构信

息应用到示例代码检索中，设计了一个基于频繁项集挖掘的 API 使用示例挖掘方法。他们首先基于现有代码搜索引擎获取相关源代码文件，然后利用频繁项集分析方法对检索结果中 API 的使用模式进行挖掘，从而分析抽取得到一个精炼的 API 使用示例。类似地，Bajracharya 等人[76]提出了一种结构化语义索引技术(Structured Semantic Index， SSI)，利用不同软件项目对同一 API 使用的相似度来建立源代码实体与不同表述词汇的关联，然后根据用户需要从代码库中检索相应 API 的使用示例代码。

3.3.3 开发知识与开发环境的集成

软件知识分享社区中积淀的大量软件开发知识如果能与开发环境集成将有助于提高开发者效率。研究人员研究实现了一系列将知识分析社区与缺陷管理工具、集成开发环境等相结合的工具和方法。针对编程开发，Alberto[74]设计了一种将 IDE 与离线 SOF 数据无缝衔接的 Eclipse 插件，该插件能够将 Eclipse 中的源代码与 SOF 中的相关讨论关联起来，并能向其中添加相应的链接。开发知识与开发环境的集成使得开发者在进行开发时不用切换场景，能够有效提高开发者的工作效率。针对软件缺陷，Correa 等[77][21]以 Chrome 为例分析了在 Chrome 缺陷跟踪工具中嵌入的外部链接的类别以及这些链接对缺陷解决所带来的影响，并聚焦于 StackOverflow 设计了一个基于标题文本相似度的方法来为缺陷推荐 StackOverflow 平台上的相关问答。

综上所述，当前面向大众化软件生产环境下的知识管理研究主要面向编程知识的管理，包括基于软件知识分享社区对代码检索、编程问题解答、API 使用等，采用文本挖掘和代码结构分析相结合展开研究，并对如何将开源社区知识集成到开发环境中进行了初步探索。虽然软件知识分享社区积累了丰富的知识，但是相比传统的软件开发文档，这些知识数据存在重复数据多、质量参差不齐等问题，当前工作尚未对此进行深入研究。同时，当前研究对知识分享社区中面向开发活动维护文档的分析和挖掘涉及较少，虽然有部分文章对缺陷和相关讨论进行自动关联展开研究，但是仅基于简单的相似度分析方法难以获得理想的效果。如何对知识分享社区中的需求知识和缺陷知识进行筛选整合，将语义分析与社交网络结合起来，实现跨社区、高质量、高准确度的项目文档自动关联及自动维护是一项极具研究价值的工作。

4 未来展望与挑战问题

近年来，软件工程领域主要围绕软件协同开发社区的仓库数据开展了大量挖掘工作，其中围绕资源定位、任务分配和知识管理的挖掘工作一直是研究热点。但这些研究主要局限于某个特定开源社区或者某个特定的软件项目，没有以更广阔、更综合的视角去研究开源社区数据的挖掘问题。从网络科学的角度来看，开源社区数据中的大规模多样化信息实体及其关联关系已经形成一 种边界开放、内涵丰富的信息网络，本文称之为**软件信息网络(software information network)**。软件信息网络本质上是一种异质信息网络，它的节点和边可具有多种类型，能够很好的将不同社区中的信息实体及其丰富的结构关系和语义关系融合起来，对其处理和分析有助于充分发掘开源社区数据中蕴含的信息和知识，为解决大众化软件生产面临的新问题提供了一种全新思路。那么，如何构造软件信息网络？如何利用软件信息网络更好的解决大众化软件生产环境下的可信资源定位、开发任务预测和社区知识推荐等新问题？为此，本文首次提出软件信息网络挖掘问题，其内涵为：如何基于分散在互联网不同开源社区的数据实现高质量软件信息网络的构造和分析，并通过对软件信息网络的挖掘来解决软件开发过程中的可信资源定位、开发任务预测、社区知识推荐等大众化软件生产面临的新问题。综上所述，本文软件信息网络挖掘问题的提出主要基于以下判断：(1) 大众化软件开发本质上是一种全球化软件工程活动，其内在线索需要在更大的范围内考察和挖掘；(2) 软件协同开发社区和软件知识分享社区的信息实体具有很强的互补性和关联性；(3) 软件信息网络有助于综合利用各类社区中广泛互联的数据。

开源社区数据中蕴含着大量对软件开发具有重要作用的软件资源、软件知识和关联关系。自从本世纪初软件仓库挖掘技术出现以来[7]，人们利用统计、图论、机器学习、语义 Web 等手段对软件仓库数据进行挖掘，不仅发现了软件开发活动的很多特有规律，也据此设计出很多新型软件工具。近年来，软件仓库挖掘已经扩展到各类开源社区的数据，呈现出快速扩展的趋势。分析已有文献，目前研究工作的局限性及其发展趋势主要表现在以下几个方面。

4.1 不同类型开源社区的数据融合与利用

根据研究数据对象的不同，现有开源社区数据挖掘的研究大致可以划分为面向协同开发社区的数据挖掘(如 SourceForge 版本库数据挖掘)和面向知识分享社区的数据挖掘(如 StackOverflow 问答数据挖掘)。软件协同开发社区逐步由集项目门户和各种开发工具于一体的综合社区逐步发展为仅包含版本、缺陷管理和社交工具的专业生产社区，其开发过程的知识分享和技术支持越来越依赖于软件知识分享社区。现有的软件仓库挖掘研究主要利用来自协同开发社区的软件版本库、缺陷库、邮件列表等数据，忽略了对知识分享社区中数据的利用。

目前，软件知识分享社区已经成为用户反馈、技术支持等软件维护活动的重要平台，其数据能够有效反映软件的流行度、应用规模、用户评价以及大量更为具体的内容。这类社区本质上是对软件在互联网范围的大众化测试结果的发布和交流平台，能够对软件仓库挖掘给以很多补充。例如，现有的软件特征定位通常会得到大量功能相似的软件包或 API，对其进一步筛选往往是一项耗时且困难的工作，但是通过考察候选结果在知识分享社区中的热度和评价则可能会得到传统方法达不到的效果。因此，两类社区数据的综合利用将是本领域一个重要发展趋势。

4.2 结构化关系和语义化关系的关联与挖掘

根据研究方法的不同，现有开源社区数据挖掘研究大体可以分为基于结构关系的挖掘和基于语义关系的挖掘。其中基于结构关系的挖掘主要是基于项目中程序调用、软件依赖、开发者协同、开发者与开发制品等之间的关联关系，利用 PageRank、随机游走等图论和社交网络分析方法进行挖掘；基于语义关系的挖掘则基于源代码标识符、代码注释、社区讨论等文档数据，利用支持向量机、随机主题模型等文本挖掘方法进行分析。

在互联网软件协同开发社区和软件知识分享社区数据中，针对同一对象，如第三方软件 API，既有软件依赖的结构化关联信息，也存在包名、方法名、代码注释、说明文档等语义化信息，两类信息从不同侧面描述了对象实体。现有绝大部分相关工作主要侧重于对结构关联关系或者语义关联关系某一方面单独展开研究，虽然有部分工作将软件代码中的文本语义关系与代码依赖的结构关系结合起来进行研究，但是这种关联也仅仅局限在项目或是社区内部，忽略了在跨社区范围内两者之间广泛存在的互补性。在信息网络研究领域，将结构化关系与语义关系融合起来已有不少研究成果。如何充分利用软件工程数据特点、根据挖掘任务将结构化关系挖掘方法和语义分析方法有机结合起来将是下一步的研究热点和难点。

4.3 面向开源社区数据的统一信息模型和框架

现有软件仓库挖掘领域的研究工作主要按研究问题的不同进行区分，尚未形成具有明晰脉络的技术体系。这一方面是因为该领域不断有新的数据类型和新问题涌现出来，数据结构庞杂、规范性差，研究工作尚处于初期的快速发展阶段；另一方面则是由于缺少从全局和共性的角度去考虑开源社区数据的信息模型及其挖掘问题的基础性研究工作。这种情况不利于该领域相似研究工作的继承、借鉴和比较，也不利于该领域的长期发展和积累。事实上，从上述相关工作的分析不难发现，很多软件数据挖掘问题都是采用 SourceForge、Github 和 StackOverflow 等几个大型数据集，而这些数据集完全可以共享同一个信息模型和基础算法。遗憾的是目前本领域没有像学术文献数据挖掘领域那样形成相对公认的理论框架。虽然学术文献数据的标准化经历了数十年时间的发展才打到目前的程度，但开源社区数据的标准化随着软件开发工具的发展越来越得到重视，加强基础性研究的基本条件已经具备。

4.4 面向大众生产的新型软件开发支撑工具

在大众生产模式下，外部贡献是大众化软件生产的主要推动力，近年来其在软件项目中的比重呈现出不断增长的趋势。尤其是随着以 Github 为代表的社交编程社区以及 StackOverflow 为代表的软件开发问答社区的兴起，越来越多的外部开发者通过 Pull-Request 方式贡献代码、通过参与问答贡献知识。围绕外部贡献有

效组织软件协同开发活动，包括开发任务的有效组织和社区软件知识的高效利用是大众化软件生产能否成功的重要前提。

现有面向软件协同开发的挖掘技术，如缺陷分类、缺陷指派和专家发现等，主要是根据开发者在项目内部的历史行为和贡献评估其开发能力。如何将开发任务和软件缺陷指派给项目核心成员以外的其他开发者，或者如何邀请具有丰富经验的外部开发者来评阅项目的外部贡献，是现有方法难以解决的新问题。其主要原因一方面在于很难获取到外围开发者的项目开发数据，另一方面即使获取到其项目开发数据也很难判断其对该项目的熟悉程度和关注程度，这些问题都需要引入新的机制和方法来解决。

此外，利用社区知识来简化开发者的维护工作，如完善需求文档、补全缺陷描述等，是目前的重要趋势。软件开发需要将分散在知识分享社区中的高质量文本与开发活动关联起来，对软件开发给以说明。如何将开发者在知识分享社区中发布的知识推荐到其在开发社区的相关开发活动，以实现更为有效的知识管理机制，是现有研究尚未深入研究的问题。

近年来，信息检索(information retrieval)领域开始从网络科学的角度研究数据挖掘问题。Han 等人将现实世界的数据集合建模为异质信息网络(heterogeneous information network)，将聚类、分类、排序、搜索、主题分析、关系预测等大量数据挖掘问题转化为信息网络分析问题[27][78]，在学术文献数据挖掘、社交网络数据挖掘等多个领域得到成功应用。我国学者也在该方向取得了丰硕的学术成果并开展了成功实践[79][80]。异质信息网络分析技术强调充分利用不同类型的信息实体及其之间的关联关系，对大规模异构数据条件下的软件资源定位、开发任务预测、社区知识推荐，以及建立统一的软件数据信息模型和研究框架具有重要借鉴意义。

5 结论

在互联网环境下，大众参与的软件开发模式作为一种新型的软件生产模式在软件开发活动中有了越来越大的生产力，产生了巨大的影响力。与之相应的，开源社区也出现了大量的软件资源和知识文本，呈现了出大数据特征，并为大众软件开发活动提供支持和帮助。同时，大众软件开发也面临着软件资源定位，开发任务预测和软件知识推荐等亟需解决的问题。本文围绕着这三个方面展开了研究，将软件仓库挖掘技术引入了软件信息网络挖掘，结合了软件协同开发社区和知识分享社区两类社区数据之间很强的互补性和关联性，突破了相关研究面向某个特定的开源社区甚至特定的软件的局限性，以更广阔、更综合的视角综合分析并总结了相关研究工作，为大众化软件生产提供了新的研究思路。最后本文指出了开源社区数据挖掘技术未来的发展趋势，即软件协同开发社区和知识分享社区的融合可望形成能够综合利用其中信息和知识的软件信息网络，而对这个软件信息网络的挖掘是信息检索领域最新理论和软件仓库挖掘技术相结合的研究思路，具有很强的新颖性和极大的研究价值，可望在大众化软件生产条件下软件开发技术的关键技术方面取得突破。

致 谢 本文得到国家自然科学基金课题“面向大众生产的软件信息网络挖掘及其应用研究”（NO 61472430）和国家重点研发计划“面向群体化方法的软件智能开发服务环境”的资助（NO 2016YFB1000805）。

References:

[1] 杨芙清, 吕建, 梅宏, 网构软件技术体系: 一种以体系结构为中心的途径. 中国科学E 辑: 信息科学 2008 年, 第 38 卷, 第 6 期: 818 -828

[2] 王怀民, 尹刚, 谢冰, 刘旭东, 魏峻, 刘江宁, 基于网络的可信软件大规模协同开发与演化, 中国科学：信息科学, 软件维护与演化技术专刊, 2014, 44(1): 1-19.

[3] Audris Mockus: Amassing and indexing a large sample of version control systems: Towards the census of public source code history. MSR 2009: 11-20

[4] Howison, J., Conklin, M., & Crowston, K. FLOSSmole: A collaborative repository for FLOSS research data and analyses.

International Journal of Information Technology and Web Engineering, 1(3), 17–26, 2006.

[5] Bogdan Vasilescu, Vladimir Filkov, Alexander Serebrenik: StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge. SocialCom 2013: 188-195

[6] Yanzhen Zou, Changsheng Liu, Yong Jin, Bing Xie: Assessing Software Quality through Web Comment Search and Analysis. ICSR 2013: 208-223

[7] Audris Mockus, Roy T. Fielding, James D. Herbsleb: Two case studies of open source software development: Apache and Mozilla. ACM Trans. Softw. Eng. Methodol. 11(3): 309-346 (2002)

[8] Kevin Crowston and Barbara Scozzi. Coordination practices within OSS development teams: The bug fixing process. In Computer Supported Acitivity Coordination, pages 21-30. INSTICC Press, 2004.

[9] Gousios, Georgios, Martin Pinzger, and Arie van Deursen. An exploratory study of the pull-based software development model. In Proceedings of the 36th International Conference on Software Engineering, pp. 345-355. ACM, 2014.

[10] Filippo Ricca, Alessandro Marchetto: Are Heroes common in FLOSS projects? ESEM 2010, Article No. 55

[11] Hao Zhong, Tao Xie, Lu Zhang, Jian Pei, Hong Mei: MAPO: Mining and Recommending API Usage Patterns. ECOOP 2009: 318-343

[12] Siddharth Subramanian, Reid Holmes: Making sense of online code snippets. MSR 2013: 85-88

[13] Jihyun Lee, Jin-Sam Kim, Gyu-Sang Shin: Facilitating Reuse of Software Components using Repository Technology. APSEC 2003: 136-142

[14] Abdallah Mohamed, Günther Ruhe, Armin Eberlein: COTS Selection: Past, Present, and Future. ECBS 2007: 103-114

[15] Jane Cleland-Huang, Horatiu Dumitru, Chuan Duan, Carlos Castro-Herrera: Automated support for managing feature requests in open forums. Commun. ACM 52(10): 68-74 (2009)

[16] Pieter Hooimeijer, Westley Weimer: Modeling bug report quality. ASE 2007: 34-43

[17] John Anvik, Lyndon Hiew, and Gail C. Murphy. Who should fix this bug? In Proceedings of the 28th International Conference on Software Engineering, pages 318-370, 2006.

[18] Swapna Gottipati, David Lo, Jing Jiang: Finding relevant answers in software forums. ASE 2011: 323-332

[19] Daqing Hou, Lingfeng Mo: Content Categorization of API Discussions. ICSM 2013: 60-69

[20] Laura V. Galvis Carreño, Kristina Winbladh: Analysis of user comments: an approach for software requirements evolution. ICSE 2013: 582-591

[21] Denzil Correa, Ashish Sureka: Integrating Issue Tracking Systems with Community-Based Question and Answering Websites. Australian Software Engineering Conference 2013: 88-96.

[22] Yu, Liguo, and Srini Ramaswamy. "Mining cvs repositories to understand open-source project developer roles." Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on. IEEE, 2007.

[23] Raemaekers, Steven, Arie van Deursen, and Joost Visser. "An analysis of dependence on third-party libraries in open source and proprietary systems." Sixth International Workshop on Software Quality and Maintainability, SQM. Vol. 12. 2012.

[24] Didi Surian, Nian Liu, David Lo, Hanghang Tong, Ee-Peng Lim, Christos Faloutsos: Recommending People in Developers' Collaboration Network. WCRE 2011: 379-388

[25] Horatiu Dumitru, Marek Gibiec, Negar Hariri, Jane Cleland-Huang, Bamshad Mobasher, Carlos Castro-Herrera, Mehdi Mirakhorli. On-demand feature recommendations derived from mining public product descriptions, Proceedings of the 33th IEEE international conference on Software and Engineering, ICSE 2011, 181-190.

[26] Xin Xia, David Lo, Xinyu Wang, Bo Zhou: Tag recommendation in software information sites. MSR 2013: 287-296

[27] Jiawei Han, Yizhou Sun, Xifeng Yan, Philip S. Yu: Mining Knowledge from Data: An Information Network Analysis Approach. ICDE 2012: 1214-1217

[28] OpenBrr, Business Readiness Rating for Open Source [EB/OL], http://www.openbrr.org, 2005-01-01/2011-10-18

[29] Atos Origin, Method for Qualification and Selection of Open Source software (QSOS), http://www.qsos.org, 2006-04-01/2011-10-18.

[30] Samoladas I, Gousios G, Spinellis D, et al. The SQO-OSS quality model: measurement based open source software evaluation[J]. Open source development, communities and quality, 2008: 237-248.

[31] Luigi Lavazza, Sandro Morasca, Davide Taibi, Davide Tosi: Predicting OSS trustworthiness on the basis of elementary code assessment. ESEM 2010

[32] Veronika Bauer, Lars Heinemann, Benjamin Hummel, Elmar Jürgens, Michael Conradt: A framework for incremental quality analysis of large software systems. ICSM 2012: 537-546

[33] Secil Ugurel, Robert Krovetz, C. Lee Giles: What's the code? Automatic classification of source code archives. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002: 639-644.

[34] Collin McMillan, Mario Linares Vásquez, Denys Poshyvanyk, Mark Grechanik: Categorizing software applications for maintenance. ICSM 2011: 343-352.

[35] Shinji Kawaguchi, Pankaj K. Garg, Makoto Matsushita, Katsuro Inoue, MUDABlue: An automatic categorization system for open source repositories, 11th Asia-Pacific Software Engineering Conference, APSEC 2004, 184-193.

[36] Kai Tian, Meghan Revelle, Denys Poshyvanyk, Using latent dirichlet allocation for automatic categorization of software, 6th IEEE International Working Conference on Mining Software Repositories, 2009, 163 – 166.

[37] Tao Wang, Huaimin Wang, Gang Yin, Charles X. Ling, Xiang Li and Peng Zou. Mining Software Profile across Multiple Repositories for Hierarchical Categorization. ICSM 2013, pp.240-249.

[38] Maximilian Steff, Barbara Russo, Günther Ruhe: Evolution of features and their dependencies - an explorative study in OSS. ESEM 2012: 111-114.

[39] Bing Xie, Meng Li, Jing Jin, Junfeng Zhao, Yanzhen Zou: Mining Cohesive Domain Topics from Source Code. ICSR 2013: 239-254.

[40] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, Patrick Heymans: Feature model extraction from large collections of informal product descriptions. FSE 2013: 290-300.

[41] Bogdan Dit, Meghan Revelle, Malcom Gethers, Denys Poshyvanyk: Feature location in source code: a taxonomy and survey. Journal of Software: Evolution and Process 25(1): 53-95 (2013).

[42] Gregory Gay, Sonia Haiduc, Andrian Marcus, Tim Menzies: On the use of relevance feedback in IR-based concept location. ICSM 2009: 351-360.

[43] Denys Poshyvanyk, Malcom Gethers, Andrian Marcus: Concept location using formal concept analysis and information retrieval. ACM Trans. Softw. Eng. Methodol. 21(4): 23 (2012).

[44] Kunming Nie, Li Zhang: Software Feature Location Based on Topic Models. APSEC 2012: 547-552

[45] Bogdan Dit, Meghan Revelle, Denys Poshyvanyk: Integrating information retrieval, execution and link analysis algorithms to improve feature location in software. Empirical Software Engineering 18(2): 277-309 (2013).

[46] Nilam Kaushik, Ladan Tahvildari, Mark Moore: Reconstructing Traceability between Bugs and Test Cases: An Experimental Study. WCRE 2011: 411-414.

[47] Alberto Bacchelli, Michele Lanza, and Romain Robbes. Linking e-mails and source code artifacts. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE '10):375-384.

[48] Nasir Ali, Yann-Gaël Guéhéneuc, Giuliano Antoniol: Trustrace: Mining Software Repositories to Improve the Accuracy of Requirement Traceability Links. TSE 2013, 725-741.

[49] Tathagata Dasgupta, Mark Grechanik, Evan Moritz, Bogdan Dit, Denys Poshyvanyk: Enhancing Software Traceability by Automatically Expanding Corpora with Relevant Documentation. ICSM 2013: 320-329.

[50] Giovanni Capobianco, Andrea De Lucia, Rocco Oliveto, Annibale Panichella, Sebastiano Panichella: Improving IR-based Traceability Recovery via Noun-based Indexing of Software Artifacts. Journal of Software: Evolution and Process, Volume 25, Issue 7, pages 743–762, July 2013.

[51] Andrew Begel, Yit Phang Khoo, and Thomas Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering(ICSE '10):125-134.

[52] Ramin Shokripour, John Anvik, Zarinah Mohd Kasirun, Sima Zamani: Why so complicated? Simple Term Filtering and Weighting for Location-based Bug Report Assignment Recommendation. MSR 2013: 2-11.

[53] Lamkanfi A, Demeyer S, Gigery E, Goethals B. Predicting the severity of a reported bug. In Proc. the 7th Working Conference on Mining Software Repositories, Cape Town, South Africa, May 2010, Pages 1–10.

[54] Lamkanfi A, Demeyer S, Soetens Q D, Verdonck T. Comparing mining algorithms for predicting the severity of a reported bug. In Proc. the 15th European Conference on Software Maintenance and Reengineering, Oldenburg, Germany, March 2011, pp.249–258.

[55] Lian Yu, Wei-Tek Tsai, Wei Zhao, Fang Wu: Predicting Defect Priority Based on Neural Networks. ADMA (2) 2010: 356-367.

[56] Kim S, Ernst M D. Which warnings should I fix first? In Proc. the 6th ESEC-FSE, Dubrovnik, Croatia, September 2007, pp.45–54.

[57] Marcelo Serrano Zanetti, Ingo Scholtes, Claudio Juan Tessone, Frank Schweitzer: Categorizing bugs with social networks: a case study on four open source software communities. Proceedings of the 2013 International Conference on Software Engineering, (ICSE 2013): 1032-1041.

[58] Hoda Naguib, Nitesh Narayan, Bernd Brügge, Dina Helal: Bug report assignee recommendation using activity profiles. MSR 2013:

22-30.

[59] Xin Xia, David Lo, Xinyu Wang, Bo Zhou: Accurate developer recommendation for bug resolution. WCRE 2013: 72-81.

[60] Jifeng Xuan, He Jiang, Zhilei Ren, Weiqin Zou: Developer prioritization in bug repositories. Proceedings of the 2012 International Conference on Software Engineering, (ICSE 2012): 25-35.

[61] Giuseppe Antonio Di Lucca, Massimiliano Di Penta, and Sara Gradara. An approach to classify software maintenance requests. In Proceedings of the International Conference on Software Maintenance, 2002:93-102.

[62] Mario Linares Vásquez, Kamal Hossen, Hoang Dang, Huzefa H. Kagdi, Malcom Gethers, Denys Poshyvanyk: Triaging incoming change requests: Bug or commit history, or code authorship? ICSM 2012, pages 451-460.

[63] A. Mockus and J. D. Herbsleb. Expertise browser: a quantitative approach to identifying expertise. Proceedings of the 24th International Conference on Software Engineering, 2002:503 – 512.

[64] Tung Thanh Nguyen, Tien N. Nguyen, Evelyn Duesterwald, Tim Klinger, Peter Santhanam: Inferring Developer Expertise through Defect Analysis. Proceedings of the 2012 International Conference on Software Engineering, (ICSE 2012): 1297-1300.

[65] David Ma, David Schuler, Thomas Zimmermann, Jonathan Sillito: Expert Recommendation with Usage Expertise. ICSM 2009: 535-538.

[66] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, Sebastiano Panichella: Who is Going to Mentor Newcomers in Open Source Projects? FSE 2012: 44.

[67] Mohammad Gharehyazie, Daryl Posnett, and Vladimir Filkov. Social Activities Rival Patch Submission for Prediction of Developer Initiation in OSS Projects. In Proceedings of the 2013 IEEE International Conference on Software Maintenance (ICSM '13). 2013: 340-349.

[68] Ioana Rus, Mikael Lindvall: Guest Editors' Introduction: Knowledge Management in Software Engineering. IEEE Software 19(3): 26-38 (2002).

[69] Miltiadis Allamanis, Charles A. Sutton: Why, when, and what: analyzing stack overflow questions by topic, type, and code. MSR 2013: 53-56.

[70] C. Parnin, C. Treude, L. Grammel, and M.-A. D. Storey, Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow. Technical Report GIT-CS-12-05, Georgia Tech, 2012.

[71] Stefan Henβ, Martin Monperrus, Mira Mezini, Semi-automatically extracting FAQs to improve accessibility of software development knowledge, Proceedings of the 2012 International Conference on Software Engineering, ICSE 2012:793-803.

[72] Edmund Wong, Jinqiu Yang, Lin Tan: AutoComment: Mining question and answer sites for automatic comment generation. ASE 2013: 562-567 .

[73] Barthélémy Dagenais, Martin P. Robillard: Recovering traceability links between an API and its learning resources. Proceedings of the 2012 International Conference on Software Engineering, (ICSE 2012): 47-57.

[74] Bacchelli Alberto, Ponzanelli L., Lanza M.. Harness StackOverflow for the IDE. Third International Workshop on Recommendation Systems for Software Engineering (RSSE), 2012:26-30.

[75] Tao Xie, Jian Pei: MAPO: mining API usages from open source repositories. MSR 2006: 54-57.

[76] S. K. Bajracharya, J. Ossher, and C. V. Lopes. Leveraging Usage Similarity for Effective Retrieval of Examples in Code Repositories. Proceedings of FSE 2010, New York, NY, USA, 2010:157 – 166.

[77] Denzil Correa, Sangeeta Lal, Apoorv Saini and Ashish Sureka. Samekana: A Browser Extension for Including Relevant Web Links in Issue Tracking System Discussion Forum. Proceedings of the 2013 20th Asia-Pacific Software Engineering Conference (APSEC), 2013.

[78] Yizhou Sun, Jiawei Han: Mining Heterogeneous Information Networks: Principles and Methodologies. Synthesis Lectures on Data Mining and Knowledge Discovery, Morgan & Claypool Publishers 2012

[79] Jie Tang, Sen Wu, Bo Gao, Yang Wan: Topic-level social network search. KDD 2011: 769-772

[80] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and Mining of Academic Social Networks. In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'08). pp.990-998.

[81] Yue Yu, Huaimin Wang, Gang Yin, Tao Wang, Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?, Information and Software Technology, Volume 74, June 2016, Pages 204-218.

[82] Yu, Y., Wang, H., Yin, G. and Ling, C.X., 2014. Reviewer Recommender of Pull-Requests in GitHub. ICSME, 14, pp.610-613.

附中文参考文献:

[1] 王青,伍书剑,李明树.软件缺陷预测技术.软件学报,2008,19(7):1565 1580. http://www.jos.org.cn/1000-9825/19/1565.htm

[3] 郁抒思,周水庚,关佶红.软件工程数据挖掘研究进展.计算机科学与探索,2012,6(1):1 31.

附录:

NLP: Natural Language Processing JS：Jensen-Shannon

SVM: Support Vector Machine LDA: Latent Dirichlet Allocation

LSI: Latent Semantic Indexing KNN: K-Nearest-Neighbor