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

面向高效软件开发的智能化群体协作*

张 建[1,2], 刘旭东[1,2], 王旭[1,2], 夏正林[1,2], 符阳[1,2], 孙海龙[1,2]

[1](软件开发环境国家重点实验室(北京航空航天大学),北京 100191)

[2](北京航空航天大学 计算机学院,北京 100191)

通讯作者: 孙海龙, E-mail: sunhl@buaa.edu.cn

摘 要: 软件开发高度依赖开发者的群体贡献,因而提高开发者群体的协作效率是提高软件开发效率与质量的重要问题.近年来,通过挖掘并利用软件大数据中蕴含的知识来提高软件开发的智能化水平已成为软件工程领域中的热点研究问题.然而,对软件开发者及其群体协作方法的研究尚未形成系统化的研究成果. 因此,本文以开发者群体为研究对象,通过深入分析开发者的行为历史数据,研究面向智能协作的关键技术,并以此为基础研制相应的支撑环境.本文首先分析总结了相关研究工作;然后,给出了软件开发者能力特征模型及其协作关系模型,并构建了开发者知识图谱;进一步以开发者知识图谱为支撑,以两个开发者推荐方法为例介绍了基于智能推荐的协作开发方法;基于以上关键技术,研发了相应的支撑工具,并构建了智能协作开发环境原型系统;最后,对本文的工作进行了总结和展望.

关键词: 智能化软件开发;大数据;群体协作;知识图谱;推荐系统

中图法分类号: TP311

中文引用格式:

英文引用格式:

Towards Intelligent Crowd Collaboration for Productive Software Development

ZHANG Jian[1,2], LIU Xu-Dong[1,2], WANG Xu[1,2], XIA Zheng-Lin[1,2], FU Yang[1,2], SUN Hai-Long[1,2]

[1](State Key Laboratory for Software Development Environment (Beihang University), Beijing 100191, China)

[2](School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract: Software development is highly dependent on the developer crowd contribution, thus it is an important issue to improve the collaboration efficiency of a team for improving the efficiency and quality of software development. In recent years, mining big software data and utilizing the knowledge contained in it to explore intelligent methods for software development is an active research topic. However, existing researches on software developer and crowd collaboration have not yet formed systematic methods. Therefore, this paper studies the key technologies for intelligent collaboration through in-depth analysis of developer behavior. Besides, the corresponding support environment is also developed basing on the key technologies to improve the efficiency and quality of software development. This study first analyzes and summarizes related work. Then, it proposes a systematic approach of analyzing developers and their collaboration which is called Developer Knowledge Graph. Next, supported by the Developer Knowledge Graph, two kinds of developer recommendation methods are introduced as examples to introduce the collaborative development method based on intelligent recommendation. Depending on the above technologies, the corresponding supporting tools are developed, and a prototype system of intelligent collaborative development environment is provided. Finally, this paper summarizes the work and prospects.

Key words: intelligent software development; big data; crowd collaboration; knowledge graph; recommender system

* 基金项目:国家重点研发计划项目 (2016YFB1000804),国家 973 计划课题(2014CB340304, 2015CB358700),国家自然科学基金项目（61421003）

Foundation item: National Key Research and Development Program (2016YFB1000804), National Basic Research 973 Program of China (2014CB340304, 2015CB358700), National Natural Science Foundation (61421003)

软件开发是以“开发者”为中心的智力密集型活动.尽管在形式化方法、人工智能等技术的推动下,部分开发任务可以实现一定程度的自动处理,然而由于软件开发的高度复杂性,大多数开发任务尚难以自动完成,开发者依然是软件开发过程中的核心要素.不仅如此,软件开发的复杂性使得大部分软件的开发难以由单个开发人员独立完成,任务之间的关联性使得软件开发通常需要大量开发者群体的共同协作.例如,据报道 Windows 7 的开发涉及到 2000 多人、共 25 个子团队的协作.特别是,开源软件、众包等基于互联网的软件开发模式涉及到的开发者数量庞大、且相互之间了解有限,建立开发者之间的高效协作成为更加迫切的需求.例如,开源社区 GitHub 拥有 2400 多万的注册用户,其中 pull request 机制是 GitHub 倡导的社会化编程的核心内容,但如果没有推荐技术的支持,pull request 得到评论的平均时间会延后 12 天[1];著名的软件开发问答社区 Stack Overflow 汇聚了 700 多万用户,但仍然有 420 多万问题（占总问题的 28%）得不到有效回答,其重要原因是缺乏针对问题进行回答者的推荐支持.总之,开发者群体在开发过程中的协作效率与效果在很大程度上影响着软件的开发效率和质量.因此,研究开发者的能力、行为及其相互影响并提供相应的协作支撑方法具有重要的意义.

图灵奖获得者 Frederick P. Brooks 在其著名的《人月神话》一书中明确指出了简单地基于“人月”来度量和计划软件开发任务的不足[2].例如,软件开发成本随着开发人数和时间发生变化,但进度却变缓甚至停滞.这充分说明仅考虑开发者数量和开发时间而忽视开发任务的复杂性以及开发者的协作效率是不可行的. 实际上,广义的开发者群体协作主要研究以开发者为中心的软件开发过程中各种要素之间的连接、交互与作用机理,以提升人与人、人与工具、人与数据、数据与工具之间的协作效率,从而提高软件开发的效率和质量.而狭义的群体协作主要指开发者群体之间的协作,一般体现在四个方面:(1)直接的沟通与交流;(2)开发任务的分配;(3)开发结果的汇聚;(4)开发资源的重用.在现有的软件开发中,开发者之间的协作多依赖于相关开发者的主观经验来进行,协作效率低,缺乏智能化的支持.例如,在开发者的沟通与交流方面,交流的对象往往仅限于开发团队内相互熟悉的人员,而团队外或者互联网上可能存在着大量潜在的协作者;在开发任务的分配上,一般由技术负责人依据对开发人员和开发任务的主观了解进行任务分配,缺乏客观准确的任务分配方法;在开发结果的汇聚上,多采用版本管理工具进行结果的聚合,这些工具假定每个开发者的贡献是可信的,在出现错误甚至恶意贡献的情况下,需要复杂的回滚等操作;在开发资源的重用方面,往往局限于开发团队内部所积累的软件资产库,而缺乏对互联网上更大范围的资源的重用支持.

针对软件开发中开发者协作面临的问题,本文以提高开发者协作效率为目标,研究智能化的群体协作方法,并研制智能化的协作开发支撑环境.其核心是通过对互联网及企业内等所积累的软件开发大数据的分析,对开发者的能力特征和历史协作行为等进行定性和定量地分析,进而面向特定的软件开发任务提供智能化的协作支持.首先,收集和汇聚大量的软件开发数据,学习开发者的能力特征,并挖掘其中隐含的协作关系,从而建立以开发者为中心的知识库.第二,在此基础上,一方面,根据开发者能力和行为等突破开发者推荐和智能的任务分配关键技术,提高软件开发中开发者“显式协作”的效率;另一方面,基于开发者关联突破软件资源(代码、问答等)智能推荐和重用关键技术,以提高基于软件资源的开发者“隐式协作”的效率.第三,在关键技术突破的基础上,研发相应的智能协作工具集,面向开发人员建立开放敏捷的自组织式协作支撑环境,实现智能任务分配和资源推荐等.

具体来说,本文的主要贡献包括:

(1) 提出了面向软件开发的智能化群体协作的研究框架,给出了其中的核心研究问题.

(2) 提出了开发者能力模型与协作关系模型,并基于互联网上收集的软件大数据构建了开发者知识图谱.

(3) 在开发者知识图谱的基础之上,针对开发者的智能推荐,面向众包软件开发和开源软件开发中的两种开发任务分别设计了相应的推荐算法,并基于真实数据进行了实验评估.

(4) 研制了一个智能协作开发支撑环境的原型系统.

本文第 1 节对相关的工作进行调研与总结.第 2 节描述了软件大数据驱动的智能化群体协作框架.第 3 节介绍了构建开发者知识图谱的核心问题.第 4 节针对众包开发与开源软件中两类开发任务分别给出了智能推荐算法.第 5 节描述了所研制的智能协作支撑环境的系统原型.第 6 节对本文的工作进行总结与展望.

1 相关工作

开发者协作一直是软件开发中的关键问题.早期研究表明,在大的软件开发项目中,开发人员花费 70%以上的时间进行相互的协作[3],同时也有研究表明在有些大型软件开发中团队性的活动占 85%以上[4].早期的软件开发活动大多局限于企业内部,因此相关的研究工作主要集中于提供如版本控制等工具来实现开发者共同开发软件.随着互联网的发展和成熟,尤其是社交媒体与社交网络[5](Social Network)广泛应用到软件开发中后,大量的研究开始集中于对基于互联网软件资源的开发者的协作分析.因此,本节一方面简要介绍基于版本控制工具的协作;另一方面重点介绍基于互联网软件资源的开发者协作分析的研究工作,包括开发者能力/贡献评估、开发者网络分析以及开发者/资源智能推荐.

1.1 企业内开发环境中的开发者协作

人是软件开发中的重要因素,然而早期的研究缺乏对开发者的分析,研究人员主要围绕如何管理软件开发者共同开发软件方面开展工作,其中的重点之一是对于软件版本变更的控制工具,开发者在此类工具的使用中形成了一定的协作关系.

在早期众多的版本控制工具中,并发版本系统[6](Concurrent Versions System,简称 CVS)是较早出现并广泛使用的版本控制工具.CVS 在软件开发领域中用于追踪开发工作和文件更改,并且允许多个开发者进行协作开发.但由于 CVS 在版本号上编排不合理等缺点,人们对其进行了改进,形成了新的工具 SVN[7](Subversion). 软件开发人员使用 SVN 来维护文件的当前和历史版本,如源代码、网页和文档.随后 Git[8]工具因其更加便捷、功能强大并且在版本的分支与合并上的支持度更好而越来越受到开发者的青睐.特别是,基于 Git 的分布式管理的特点,能够更好地协调多个开发者的开发工作.

此外,针对软件的缺陷管理,企业中大多使用缺陷跟踪系统 Bugzilla 等来管理软件开发中缺陷的提交、修复与关闭等整个生命周期[9].开发者在这种活动中形成了一些简单的协作,但面临着指定合适的 Bug 修复人员等问题.在整个软件开发过程中,开发者交流依赖于 Email 或即时通讯工具,这在软件开发周期中占了很大一部分比重.因为这种交流协作是一种松散的方式,很难形成统一的协作开发模式,其中的开发知识也难以被复用.

可以看出,早期的开发者之间的协作是基于管理、交流工具的.这些工具更关注软件本身,对开发者缺乏一定的重视,因此没有形成系统化、智能化的协作开发模式.

1.2 互联网环境下的软件开发者协作

在互联网技术逐渐成熟,尤其是在 GitHub 提出社交编程的概念后,以开源软件开发与软件知识共享社区为主导形成了新的开发者协作模式.研究人员开始对这种新型模式进行研究,包括开发者协作分析、开发者能力度量以及在此之上的开发者、开发资源智能推荐.

首先是开发者协作的形成与影响因素分析. Hahn 等人[10]最早对于开源软件开发(Open Source Software Development,简称 OSSD)中开发团队是如何形成的进行了探索.通过分析开源软件项目,发现当开发人员与其发起人存在强有力的合作关系时,更有可能加入该项目.随后研究人员对影响开发者协作的因素进行了一些深入的探讨,如 Roberto 等人[11]对项目中产生的 Bug 数量与开发人员的交流频率进行了相关性分析,得出了 Bug 数量与开发者交流次数正相关的结论.而 Liu 等人[12]则从集体交互的角度对影响开发者的协作行为的因素进行了实证研究.通过对比发现,开发者的经验是影响潜在协作的重要因素. Hattori [13]在先前研究的基础上,通过一种更好的方式向开发者提供相关信息以提高开发者的团队意识,从而促进协作. Blincoe [14]提出了一种及时检测协作需求的方法,即当软件项目中出现工作依赖时,反馈给开发者以使其及时展开协作.

随后,网络化分析方法成为开发者协作分析的主要手段.受社交网络的启发,人们提出了开发者网络(Developer Network)的概念[23],基于该网络解决了诸多软件演化中的问题.由于各种开发活动都与开发者息息相关,挖掘开发者网络对于更好地理解或完成开发任务起到促进作用. Meneely 等人[24]通过代码流失信息(Code Churn Information)建立开发者网络,从文件级别预测软件故障.经过实验验证,他们发现基于文件的开发者网络

度量和故障之间存在显著的相关性.其他关于开发者网络的研究与应用如文献[25]也与之类似.

社交软件工程(Social Software Engineering)则是网络化分析方法的进一步延伸.Zanetti 等人在这方面做了很多代表性的如文献[26~29]等工作.基于活跃度较高的开源软件社区如 Eclipse、Linux 等社区的数据,通过研究这种协作型的软件工程过程的社交结构与动态,探究在软件开发团队中组织协作或分配任务的方式对软件项目开发的影响.其贡献在于两方面:一是经验性地揭示了软件开发项目中协作的拓扑结构与个人或团队开发效率的关系.例如,通过研究 Gentoo,发现协作关系核心结构的变化会对软件项目构成严重威胁,团队中的核心开发者的离开导致软件开发效率严重且持续下降[31,32].二是从协作关系的角度研究协作工具,如基于软件开发社区的协作结构,研发了缺陷报告质量的自动评估工具[30].

事实上,仅从宏观层面对开发者协作的分析并不总能有效提高开发者的协作效率.为此,研究人员也从微观层面做了大量的研究,主要是对开发者的评估.

研究人员希望通过深入挖掘开发者的历史开发信息以刻画其开发贡献或开发技能.开发者在开发软件系统的过程中提升自身的专业能力,刻画这种能力对于开发任务的分配能够提供有效的信息.研究人员最早通过抽取开发者编写的源代码中的领域概念(Domain Concept)来代表开发者的技能.例如 Sindhgatta 等人[15]通过聚类源代码文档和识别每个集群中的关键区分词来提取关键术语.这些术语组成的文档与开发人员相关联,以确定开发人员擅长的技术领域.其他针对代码库的开发者评估方法[16~20]本质上与从代码文件中提取专业词汇的方法相一致,这里不再一一列举.随着社交编程的进一步成熟,近几年研究人员[21,22]开始探索更加全面的评估开发者能力的方法,主要研究开发者在问答社区(如 Stack Overflow)和开源软件社区(如 GitHub)中的开发语言能力建模. 最近,Wang 等人[44]针对众包软件开发,研究了 Topcoder 平台中开发者的技能学习曲线,刻画开发者能力随时间的动态变化,并将其用于众包开发者的推荐.

研究开发者推荐的主要目的是能够找到与不同的开发任务相匹配的开发者. 如面向缺陷修复任务,Wu 等人[33]提出了一种称为 DREX 的方法用于开发人员推荐.该方法首先使用最近邻居搜索类似的缺陷报告,然后使用开发人员的关于讨论类似的缺陷报告的历史记录来评估开发人员的专业知识.基于这两个步骤,对于新的缺陷处理任务,DREX 能够自动地推荐出合适的开发人员处理该缺陷.文献[34~36]又分别面向众包软件开发、开源软件项目以及代码评审任务做了推荐方法的研究.

此外,研究如何有效地为开发者推荐软件开发资源也是研究人员一直在探索的方向之一,其目的与开发者推荐基本相同.资源推荐的研究以代码推荐为主. 例如,Holmes 等人[37]描述了一种在代码示例存储库中定位相关代码的方法,该方法启发式地将正在开发的代码的结构与示例代码相匹配.他们从代码中自动提取查询存储库所需的结构上下文,从而节省了开发人员学习查询语言或者去编写特定代码的时间.其次,还可以从现有应用程序轻松地生成存储库. Nguyen 等人[38]研究了 API 代码的推荐. 他们提出了一种称作 APIREC 的推荐方法,能够更好地预测重复代码的更改,为开发人员提供相关的 API 建议.

与资源推荐类似的研究工作有 Web 服务的推荐,考虑用户与 Web 服务之间的关联关系,基于协同过滤的方法实现 Web 服务推荐.例如,Chen 等人[39]提出了一种用于大规模 Web 服务推荐的协同过滤算法,考虑了 QoS 与用户相关的特点,提升了 Web 服务推荐的准确率.类似地,Zhang 等人[40]也提出了基于协同过滤的 Web 服务推荐框架.所不同的是,他们考虑了 QoS 的时间特性与动态的上下文信息,并且利用张量分解来处理用户-服务-时间模型的三元关系,得到了很好的推荐结果.

从以上分析中可以看到,基于版本控制工具的协作是一种机械式的、表面性的协作,更注重于软件的变更,本身不具有智能协作的特点.而基于互联网软件资源的协作则更多的关注开发者,围绕开发者能力、关系网络,衍生出多种智能服务.然而,现有的研究大多只关注于某一个方面,而无法提供一个完整的智能协作的环境.本文提出的开发者分析方法综合了开发者能力建模与开发者协作网络,形成更加全面的开发者知识图谱,意在增强开发者协作的智能性.此外,围绕该知识图谱,本文探索了智能协作支撑环境与服务的研究.

2 智能协作框架

本文在现有研究的基础上,给出了智能协作支撑环境的框架.将多源异构的软件开发者协作开发产生的大数据以知识图谱的形式存储,并且借助于搜索、推荐等技术再把这些知识以一种智能的方式释放出来,从而实现软件开发者的智能协作.整体的研究框架如图 1 所示.

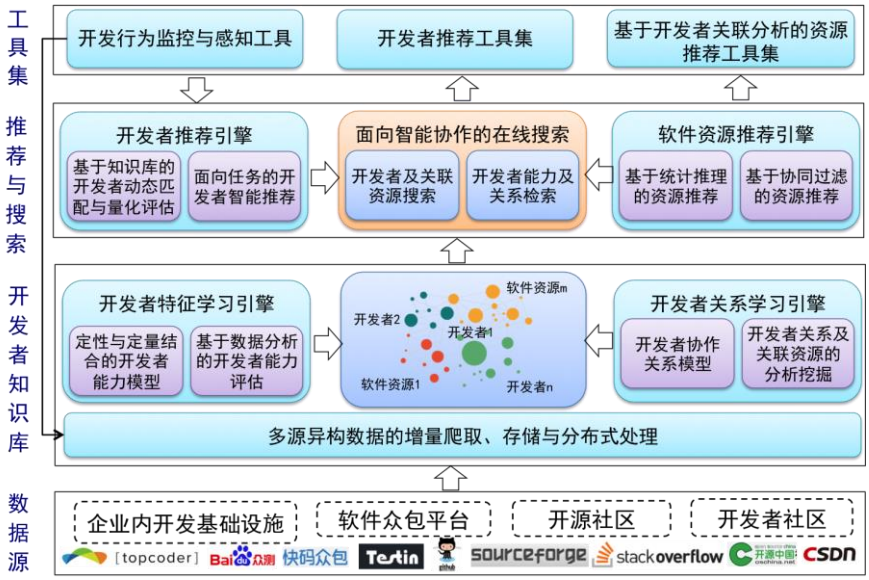


Fig. 1 The overall framework of intelligent crowd collaboration for productive software development

图 1 面向软件开发的智能化群体协作框架

在该框架中,开发者相关知识是智能协作开发环境的关键所在.具体来讲,围绕开发者知识,需要解决四类关键技术问题.

(1) 多源的软件大数据的获取与管理.从数据源增量爬取到的数据是多源异构的,因此这种原始数据包含了杂乱无章的信息.这些信息对于推荐或搜索层来说价值很低,所以不能直接使用,需要对这些数据进行有效管理和知识提取.

(2) 开发者知识库的构建.考虑到上层的软件开发者知识需求,知识提取的技术主要包括两种:开发者能力评估和关系挖掘技术.开发者能力评估技术的核心是建立定性定量结合的多维开发者能力模型,以全面地刻画开发者的能力;开发者关系挖掘技术则主要依赖协作关系模型,包含开发者-开发者关系以及开发者-软件资源的关联关系.于是构成了开发者能力、开发者关系以及开发资源的知识统一体,其表现形式为开发者知识图谱,将在下文中具体介绍.

(3) 基于开发者知识的搜索与推荐.知识存储是为了能够更好地得到智能释放,而释放的方式包括面向智能协作的知识搜索以及知识推荐.其中,知识搜索技术主要是指开发者及其关联资源搜索和开发者能力与关系检索.知识推荐是借助于开发者能力模型知识和协作关系模型的开发者推荐与开发资源推荐技术.

(4) 基于开发者知识搜索和推荐技术的相关工具集,主要包括开发者推荐和开发资源推荐工具.并且,在开发者使用这些工具进行软件开发的过程中,通过开发行为监控与感知工具,又能够扩充或更新开发者知识图谱中的知识.

在开发者知识的驱动下,这四种关键技术问题形成了闭环的智能协作环境,即数据获取→开发者知识库的构建→知识搜索与推荐→智能释放工具集→开发者知识图谱的扩充这种良性循环.

因此,智能协作的核心是如何利用开发者知识.本文提出了一种新的智能协作分析方法,即开发者知识图谱(Developer Knowledge Graph).结合了开发者能力建模和开发者协作分析的方法,形成了一体化的开发者协作知识网络,贯穿了整个智能协作环境.

3 开发者知识图谱

开发者知识图谱是以开发者作为知识节点,开发者协作关系作为知识链接的图结构的软件开发知识库.其中,节点描述了开发者的能力特征,而边描述了开发者之间的协作关系,因此定义开发者能力模型和开发者协作关系模型是构建知识图谱的关键.

3.1 开发者能力模型

从相关工作的分析中可以发现,现有的开发者能力存在三种缺陷:第一,大多数的能力评估方法只注重于对开发者的开发语言技能的抽取与评分,具有一定的片面性.第二,没有综合考虑开发者在开发活动中表现出的个人贡献与团队合作能力.第三,开发者的能力不仅体现在个人编写的代码中,在不同的开发活动中如问答、测试等也体现了其个人价值.因此,如何提供一种通用的模型来刻画各类型开发者的能力也是需要考虑的一个难题.

针对以上问题,本文提出了一种新的开发者能力模型,以多重维度考量开发者的能力,具体如图 2 所示.

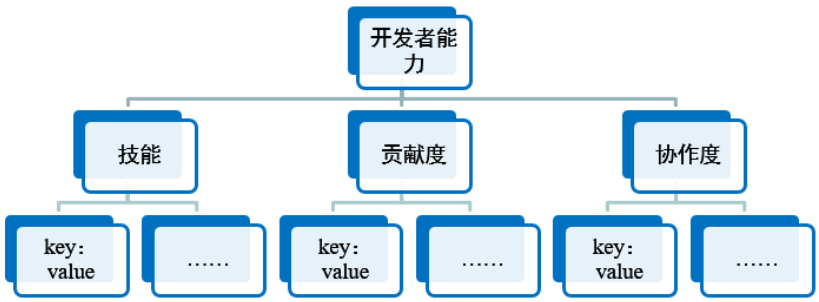


Fig. 2 The capability model of software developers

图 2 软件开发者能力模型

本文参考先前的开发者度量方法[41~43],凝练出刻画开发者能力的三个主要方面,并且考虑度量的灵活性,形成了开发者能力树.具体来说,树的根节点为开发者的总体能力评估或得分,其子树分别为专业技能(Skills)、贡献度(Contributions)以及协作度(Collaboration)三个维度.本文对于能力的刻画以统计方法为基础,特别适用于大型社区(如 GitHub、Stack Overflow)且可解释性较强.下面分别对这三个维度作具体的说明.

其中,专业技能表示开发者掌握的开发技能,包括编程语言、技术框架以及操作系统等.主要考察开发者在开发活动中是否具备某项技能以及能否符合任务需求,因此,以 key-value 的形式表示.这里 key 和 value 分别以定性和定量的方式表示开发者的专业技能掌握程度.例如,对于掌握 Java、python 语言以及 Spring 框架并且熟悉 Linux 的开发者,其专业技能可以表示为{Java:6,python:4,Spring:6,Linux:2}的集合的形式.在该集合中,key 为 Java 的技能其评分为 6,在某种评分体系下表示其能够较熟练地掌握 Java 语言,其他技能以此类推.关于技能的抽取和评分,可以集成先前的研究工作的方法.本文对于活跃在不同的社区的开发者构建了简单易行的评分体系.例如,在 Stack Overflow 社区中,从开发者回答的问题中抽取出技能标签集合,记为Γ.对于每个技能标签$T \in \Gamma$,找出开发者在该标签下的回答集合$A = \cup \alpha_i$,每个回答有对应的回答得分s_i,则开发者在该技能 T 下的技能得分为$S_T = \sum_{\alpha_i \in A} s_i$.同样地在其他社区中也有类似的计算方式,这里不再赘述.

贡献度则是指开发者在特定社区中对该社区所做出的贡献, 主要考察开发者在社区中的活跃程度,同时也是对其熟练度的刻画.这些贡献便指的是其各项活动,如在 GitHub 开源社区中参与项目或提交代码、在 Stack Overflow 问答社区中回答问题等.为了客观地度量各个社区中的开发者的贡献,我们首先抽取中开发者的不同贡献的集合$\mathrm{C}=\{C_i\,|i=1,2,3\ldots N\}$组成该开发者的贡献的 key 集合,对于每一个 key 即C_i,以属于该 key 的活动集合的累加数目V_i组成在该 key 下的贡献值.即$V_i=\sum_{k\in C_i}$.例如,开发者 D 在 GitHub 中提交了 m 次代码,则其贡献度表示为{commit:m},其他贡献也是同样的计算方式.

协作度指开发者在特定社区的开发活动中与其他开发者的关联关系,用于考察开发者与其他开发者之间的团队协作能力.开发者在特定社区中共同完成任务或者互相关注,形成了诸多关系,主要包括社交关系与协作关系.例如,在 GitHub 中,开发者可以关注(follow)其他开发者,也可以被关注,因此产生了较为紧密的社交关系.由于每个社区的协作方式存在差异,在计算协作度时仍然按照统计协作频度的方式对开发者的协作能力进行评价,以保持模型的通用性.

按照上述方法,能够得到每个维度下的详细评分.对于这些详细评分,可按照该子树的分支的权重进行加权计算得到整个维度的评分.以某个开发者的 Skills 维度为例,假设其中包含 N 个技能标签$T_1,T_2\ldots,T_N$,对应的得分为$S_1,S_2\ldots,S_N$,按照某种任务需求其技能权重为$\omega_1,\omega_2\ldots,\omega_N$,则其 Skills 的整体得分为

$$\mathrm{P}_1=\sum_{i=1}^{N}\omega_i S_i$$

同样地可得到 Contribution 和 Collaboration 维度下的得分,记为P_2、P_3.考虑不同的侧重点,每个维度又可以有权重μ_1,μ_2,μ_3.于是,该开发者的总得分为

$$\mathrm{S}=\sum_{j=1}^{3}\mu_j\,\mathrm{P}_j$$

这项得分评价了开发者的整体能力.

从上文可以看出,本文提出的开发者能力模型具有以下几种特点:

(1) 多面性.本文在研究了先前的建模方法后,综合考虑了评估开发者能力的多项指标,解决了现有方法存在的片面性问题.

(2) 定性与定量结合.即在考虑开发者会何种技能或者贡献了哪些内容的同时,也分别对其进行了客观的量化处理,这种定性与定量分别对应于各个维度下的 Key 与 Value.

(3) 可扩展.对于本模型的第三层来说,其数量并非是固定的,而是以开发者的自身属性为准.每个开发者掌握技能或贡献的多少有统一的表示.

3.2 开发者协作关系模型

开发者协作关系分析与开发者能力评估同等重要,它们都是知识凝练的结果.从宏观的角度来看,本文的开发者协作关系的表示方法与开发者网络是相似的.然而现有的研究提出的开发者网络仅仅是开发者协作关系模型的一部分,因其往往只分析了某一种协作关系如共同修改某一文件等.另外,对于协作关系强度的度量也被淡化,从而缺失了一些关键信息.

经过综合分析,本文首次提出了一种开发者协作模型,即抽象出的跨社区的协作关系模型,该模型在各个社区中均是适用的.其在知识图谱中表示为开发者节点之间的边,这些边包含的协作关系知识如图 3 所示.

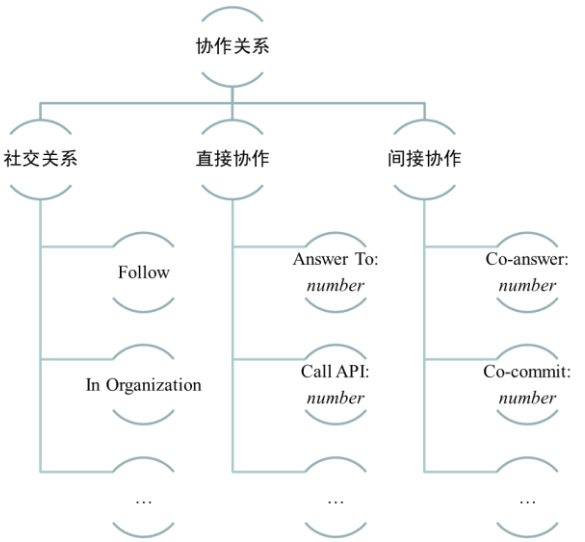


Fig. 3 The collaborative relationship model among software developers

图 3 软件开发者协作关系模型

按照协作关系类型,将所有的协作关系划分为三大类,分别是社交关系、直接协作与间接协作.在每个类型下又包含了详细的协作关系,用关系类型和关系强度作为协作关系的属性.

在开发者的开发活动中,社交关系是指开发者之间的在特定社区中的人际关系.虽然与开发活动并不直接相关,但是社交关系有时决定了软件开发的效率,因此它也是协作关系的一部分.从个人和集体的两个方面,社交关系又可以分为Follow关系和In Organization关系,分别表示开发者之间的互相关注以及开发者共同加入组织的行为.

直接协作指开发者之间的直接交互关系,即两个开发者面对同一任务需要紧密的沟通合作.例如开发者D_1与开发者D_2之间存在 Answer To 和 Call API 的关系,则分别表示D_1回答了D_2提出的问题以及D_1实现某些功能时调用了D_2提供的接口,number 则表示这些协作的频度.此外,开发者共同修改代码文件、对于同一代码的开发和测试也属于这种紧密型的协作关系.其他类似的协作关系还有很多,不再一一例举.

间接协作指开发者之间的间接交互关系.它相对于直接协作来说是一种较弱的协作关系,但对于整体的开发任务也起到了一定的作用.例如开发者D_1与开发者D_2都回答了某个问题,或者都向某个开源项目提交了自己的代码,则他们之间分别构成了 Co-answer 和 Co-commit 的关系.同样,这里 number 用来表示间接协作的关系发生的频度.类似的协作关系还有众包社区中开发者共同参与某一项目中的不同任务等.

以上描述的开发者协作关系模型,是一种简单直观的关系分析方法的结果,也是开发者知识图谱构建的关键环节.

4 基于开发者知识图谱的智能推荐方法

在丰富的历史软件开发数据的支撑下,同时借助于对开发者建模分析形成的开发者知识图谱,可提供两种开发者推荐与开发资源推荐两种智能服务.

(1) 开发者推荐.针对当前软件团队组织的相对固化和依靠主观经验、准确性不高的问题,解决的核心是根据任务特征实现开发者的动态检索和智能推荐.因为开发任务具有自己的特性,开发者能力是复杂多维的,开发之间存在关联依赖,所以开发者的智能推荐具有个性化需求、复杂度高的特点.在本节中,本文将介绍面向编码、

测试以及问答等任务特征建立需求模型,通过和开发者能力、关系网络进行动态匹配和评估,从而实现的开发者的智能推荐和任务分配.

(2) 开发资源推荐.软件开发社区中的开发资源如代码、问答等具有海量、可重用的特点,但也是由于这些资源语义丰富、结构复杂、数据量大,使得已有的推荐方法准确度不高、性能较低,同时资源的检索质量也较差,对开发者获取开发资源带来了很多困难.本文认为基于开发者知识图谱,从资源与开发者的关联关系出发,利用协同过滤的方法并结合开发环境上下文,能够提升软件资源推荐的效果.

本文重点围绕众包和开源两类开发模式分别介绍一种众包开发者推荐算法[45]以及针对 GitHub 中 pull request 的 reviewer 推荐方法[46].

4.1 面向众包软件开发任务的开发者推荐

在众包软件开发过程中,开发者和任务的不匹配不仅使得任务完成质量较低,而且也不利于开发者能力的提升.在 Topcoder 众包平台上,开发者和任务不匹配问题尤为突出,一方面众包平台发布任务过多,开发者花费过多时间精力寻找匹配任务;另一方面大量的开发者存在无提交行为,而且大量的提交质量较低,导致了众包任务完成质量较差.因此,本文研究设计了一种针对众包任务的开发者推荐方法,为任务推荐最佳匹配的开发者.

该方法由基于聚类的分类算法和关联关系分析算法两部分组成.首先对众包任务进行了特征分析,提取了任务的时间、报酬、需求描述、技能、平台等重要特征并构造了特征向量,采用聚类算法对任务进行初步分类.该聚类算法考虑不同任务之间的内容相似性以及时间局部性特点,进一步在相似任务间建立分类器,对开发者进行预测和初步筛选.

此外,该方法分析开发者之间关联关系,并建立开发者关系网络,进一步计算开发者之间的关联关系强度.一个开发者协作网络的实例,如图 4 众包开发者协作关系网络示例所示,每个节点代表开发者,节点之间的边以及边的强度代表其协作关系,其中W_1和W_2的协作关系为W_1在历史开发中赢过$W_2$10 次,输过 2 次.

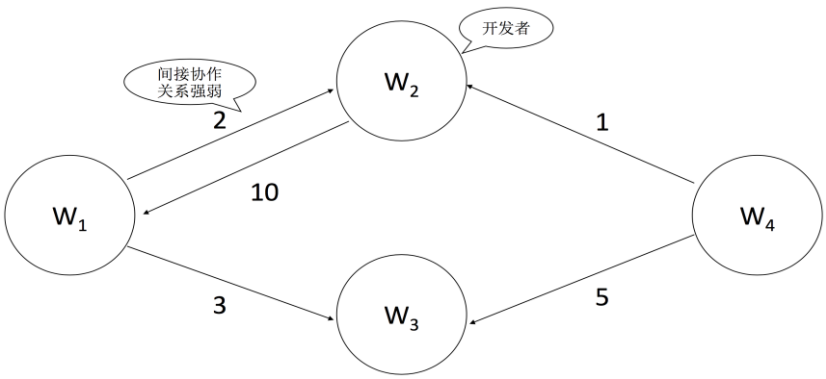


Fig. 4 An example of collaborative network among crowdsourcing developers

图 4 众包开发者协作关系网络示例

最后利用关系网络来增强推荐准确性,关联分析算法具体内容见算法 1.在该算法中,首先根据历史众包任务中开发者之间的竞争关系建立了开发者之间的网络（第 11-13 行）,具体的关系网络如图 4 所示,然后基于该竞争关系网络定义并计算了开发者之间的“吸引”关系 attraction(di,dj)= $\frac{\text{edge(di,dtj)-edge(dtj,di)}}{\deg(di)}$(第 14-16 行),该关系一方面体现了开发者之间能力水平高低,另一方面体现了开发者之间的竞争频繁程度,所以该吸引关系越强表明开发者有更高获胜的可能性.因此,计算出开发者之间的竞争关系后,可以将更高“吸引”力较强的开发者调整到推荐候选集前面(第 17-21 行),可以优化开发者的推荐顺序,提升推荐效果.

在实验过程中,实验数据来源于 Topcoder 众包软件平台,通过 Topcoder 官方 API 获取了 7000 多个开发任

务,经过数据预处理后我们选取了包含任务数量最多的三种类型数据集(分别为 Code,Assembly,First2Finish 三种类型的开发任务),并基于该数据集进行了实验验证,选取了基于内容的推荐算法(CBM),协同过滤算法(CBR)进行了对比验证,推荐评价指标选取了推荐准确率（Accuracy）.分别推荐 1-5 个开发者,详细实验结果如表 1 所示.

Algorithm 1 Matching Result Re-Rank Algorithm

算法 1 匹配结果重新排序算法

Algorithm 1 Matching Result Re-Rank Algorithm

Input:
The initial top n developers D = {d1,d2,…, dn}
The competitive network G = (V;E)
Output:
Final recommended developers D'= {d1, d2,….dn}
1: D' ← ∅
2: for each developer Di ∈ D **do**
3: count the ri, si,wi respectively
4: **end for**
5: compute avgsub, avgwin
6: **for** each developer Di ∈ D **do**
7: **if** si=ri < avgsub and si=wi < avgwin **then**
8: remove di from D
9: **end if**
10: **end for**
11: **for** each developer di ∈ D **do**
12: set deg(di) ← $\sum_{j=1}^{n}$ edge(di, dj)
13: **end for**
14: **for** each developer di ∈ D **do**
15: compute attracter(di) ← {dt1 ,dt2 ,….., dtk}
satisfying max$\sum_{j=1}^{k} \frac{\text{edge(di,dtj)}-\text{edge(dtj ,di)}}{\deg(di)}$
16: **end for**
17: **for** each developer di 2 D do
18: **if** di =2 D0 then
19: add di and attracter(di) to D0
20: **end if**
21: **end for**
22: return D'

实验结果表明该算法在三组数据集上都取得了最佳的推荐效果,相比 CBM 准确率提升了约 7.6%,相比 CBR 提升了约 27.5%.此外,与 CBC(无关联关系分析)对比表明开发者之间的关联关系能增强推荐的准确性,所以该实验结果表明开发者关联关系也与任务的质量息息相关.

Table 1 The experimental results of crowdsourcing developer recommendation

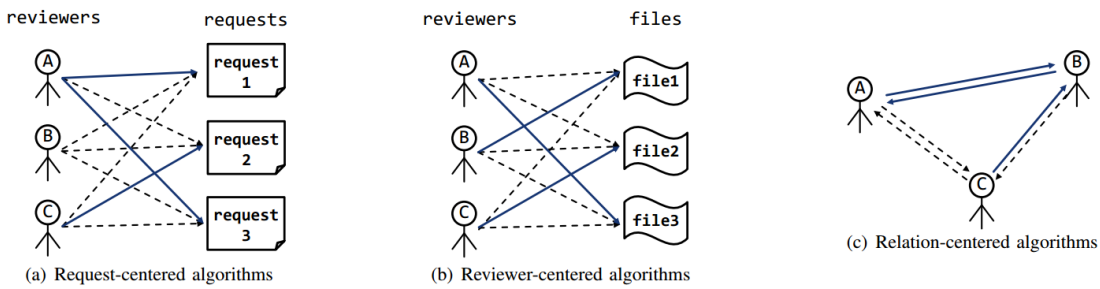
表 1 众包开发者推荐的实验结果

Dataset	Top	CBM	CBR	CBC	CNCCN
First2Finish	1	24.1	8.5	**29.1**	**29.1**
	2	32.6	13.7	**46.4**	**46.4**
	3	36.9	17.6	**52.0**	51.6
	4	45.4	23.5	**56.2**	**56.2**
	5	48.3	28.7	58.8	**59.2**
Code	1	26.6	11.5	28.8	**30.1**
	2	37.8	18.6	37.2	**40.7**
	3	41.6	26.3	42.9	**48.1**
	4	45.5	29.5	50.0	**51.9**
	5	50.6	35.3	54.5	**60.9**
Assembly	1	37.8	17.7	40.3	**42.3**
	2	48.6	21.1	50.7	**52.8**
	3	54.1	27.1	58.3	**61.4**
	4	58.6	34	63.5	**66.3**
	5	63.8	39.9	68.4	**68.6**

4.2 面向开源软件开发中代码评审任务的评审者推荐

开源软件的开发已经成为了软件开发重要的一部分.由于开源软件开发的一些不同于传统软件开发的特征,使得关于开源软件开发的研究近些年来在学术界比较活跃.考虑开源软件开发中的开发者能力模型以及开发者协作关系模型有着特别的意义.在开源社区 GitHub 中,由于缺乏在代码评审方面合适的组织工作,一些提交请求(Pull Request,简称 PR)长时间没有得到评审,一些项目核心开发者疲于应付大量的 PR.这些问题影响了项目的开发进度.准确有效的评审者推荐可以给每个 PR 提供合适的评审者,加快被评审的进度,减轻项目核心开发者的工作负载.

本文的算法主要考虑 PR 的特征与评审者之间的联系.利用协同过滤相关技术,度量 PR 特征与评审者之间的联系.具体来说分为三类联系,PR 作者与 PR 评审者之间的联系,相似 PR 与 PR 评审者之间的联系,代码文件与 PR 评审者之间的联系,如图 5 所示.



(a) Request-centered algorithms (b) Reviewer-centered algorithms (c) Relation-centered algorithms

Fig. 5 Analysis of relevance and collaboration related to the reviewer

图 5 Reviewer 相关的关联与协作关系分析

为了能够利用协同过滤的技术分析上述三类联系挖掘合适的评审者,本文将数据进行整理,形成了三个矩阵,分别是评审者-作者矩阵,评审者-PR 矩阵以及评审者-文件矩阵.在这样的数据组织形式下,利用邻居模型和隐因子模型挖掘矩阵中所蕴含的局部相似关系和整体结构信息.通过融合了这两种模型,推荐算法能够更好地推荐出合适的评审者. 具体见算法 2.

算法 2 中从第 6 到第 16 行计算了每一个开发者的能力值.该能力值主要有两部分组成,分别是矩阵中显式关联的量化值和隐式关联的计算值.算法第 11 到第 15 行负责这两部分值的计算.显式关联的计算较为直接,在矩阵构建阶段就已经确定.隐式关联的量化值的计算是由上述的融合模型完成的.最后算法 2 会依据每个开发者的能力值进行排序,排名前 K 的开发者将会被推荐作为评审者.

为了验证本文算法的有效性,Github 平台上的 5 个活跃项目的开发数据被收集,包括 issue、pull request、评论、commit 等相关信息.实验过滤了每个项目中评审者少于 2 个以及描述文本少于 5 个词语的 PR.在此基础上,实验将每个项目的最新的 600 个 PR 作为测试集,其余 PR 作为训练集,进行了实验验证. 最终实验结果如图 6. 其中的 PR-CF 是本文提出的算法,其余 4 类算法是现有研究工作的算法.在实验结果中可以看出,本文的算法在准确率上和召回率上均有较好的表现.这说明了挖掘隐式关联对于评审者推荐具有显著效果. 也说明了基于开发者知识图谱中的协作关系分析方法,与现有的推荐算法相比其推荐效果得到了较大的提升.

Algorithm 2 Reviewer recommendation algorithm

算法 2 Reviewer 推荐算法

Algorithm 2 Reviewer Recommendation Algorithm

Input:r:a new review request, k:the length of recommendation list
Output:C:a list of reviewers
1: **function** ReviewerRecommendation(r, k)
2: Construct matrix *Mat*
3: Get all reviewers as a set *reviewerSet*
4: $C \leftarrow \emptyset$
5: $colSet \leftarrow transform(r)$
6: **for** $reviewer \in reviewerSet$ **do**
7: i:Get label of $reviewer$ in $reviewerSet$
8: $score \leftarrow 0$
9: **for** $col \in colSet$ **do**
10: j:Get label of col in $columnSet$
11: **if** $Mat(i,j)$ is not null **then**
12: $score \leftarrow score + Mat(i,j)$
13: **else**
14: $score \leftarrow score + eval(i,j)$ /*catch implicit relations*/
15: **end if**
16: **end for**
17: $reviewer.score \leftarrow score$
18: $C \leftarrow C \cup \{reviewer\}$
19: **end for**
20: Sort elements in C by score of every reviewer in descending order
21: **Return** the top k elements in the C as a list
22: **end function**

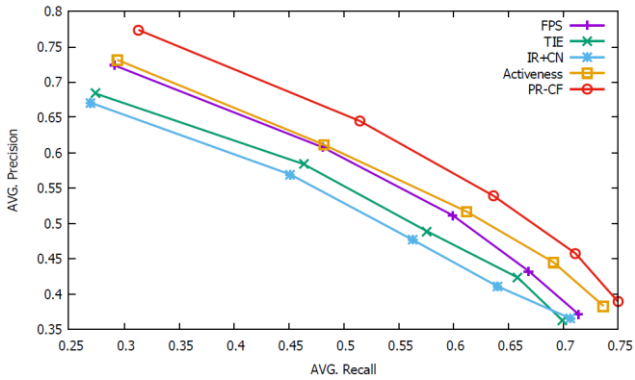


Fig. 6 The experimental results of reviewer recommendation

图 6 Reviewer 推荐的实验结果

5 智能协作开发环境原型系统

在上述研究的基础上, 本文研发了开发者能力特征和关系知识库以及相应的智能协作工具集,构建了基于开发者关联分析的智能协作开发环境的原型系统.目的在于将本文的开发者知识图谱与智能推荐集成到统一环境中,为开发人员提供更便捷的一体化服务.该系统的整体框架如图 7 所示,分为数据源、知识库、算法与服务四层.下面将围绕这四层进行介绍.

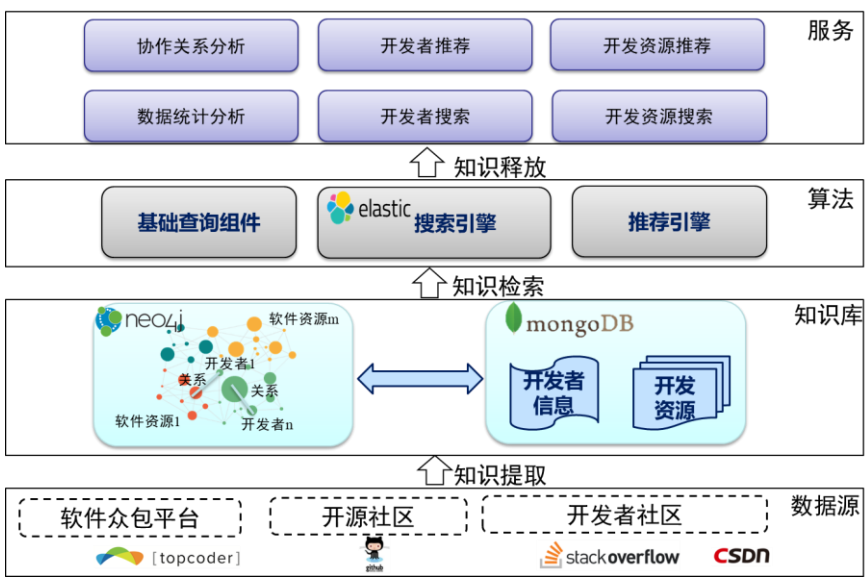


Fig. 7 The framework of intelligent collaborative development service system

图 7 智能协作开发环境的系统架构

5.1 数据源层

针对互联网中的特定开发活动,本文选择了众包开发社区 Topcoder、开源软件社区 GitHub、软件问答社

区 Stack Overflow 以及中国开发者社区 CSDN 作为数据源,以 API、直接下载、网络爬虫等方法获取了上述四个社区的原始数据,面向不同的开发任务,获取了大量的多源异构的开发者与开发资源数据,其总量约 13.2TB.

5.2 知识库层

知识库是指从原始数据中经过知识提取得到的软件开发者知识数据,包括开发者知识图谱以及与开发者相关联的开发资源.为了能够将开发者与资源建立可高效查询的关联关系,本文使用 Neo4j 图数据库来存储开发者知识图谱和开发资源节点的元数据与关联关系,用 MongoDB 来存储在第 3 节中介绍的开发者能力信息,同时开发资源的原始信息也以 MongoDB 来存储.并且在这两种数据库之间作开发者/开发资源的映射,使之相匹配,在查询时能够快速展开具体信息.本文目前构建了十分庞大的开发者知识库,包括约 9.5 亿开发者与资源节点、30.6 亿条关系.为了减轻知识搜索的压力,本文按照社区将其分为四个部分存储到不同的库中.

5.3 算法层

算法层包括基础查询算法、搜索算法以及推荐算法三个部分,从知识库进行知识检索以支撑上层服务.

第一部分的基础查询算法指直接从数据库中读取数据并做基本统计、排序的算法.如在数据统计时,需要查询 MySQL 数据库并对结果做统计处理以获取各种类型的数据总量.同样地,在获取协作关系时,需要查询 Neo4j 数据库以得到不同节点之间的关系信息.

第二部分的搜索算法包括开发者搜索以及开发资源搜索.其中,开发者搜索又分为基本搜索以及能力搜索.基本搜索是指根据开发者的基本信息搜索,如根据其用户名、邮箱名等字段搜索,是一种简单的多字段搜索.能力搜索是根据输入的能力标签来搜索出符合条件的开发者,并且基于搜索出的开发者能力评估信息来对这些开发者进行排序.

为了能够以统一、快速的方式搜索出符合某些能力条件的开发者,本文对每个开发者的能力信息进行了扩充,将能够代表该开发者能力的文档整理成他的技能详细文档.例如,对于 Stack Overflow 的开发者,其扩充文档为该开发者回答的投票数大于 0 的所有答案组成的文档.问题能力搜索的流程如图 8 所示,本文基于 ElasticSearch 简称 ES 来实现建立技能文档索引、关键词-文档匹配以及辅助排序整个流程.

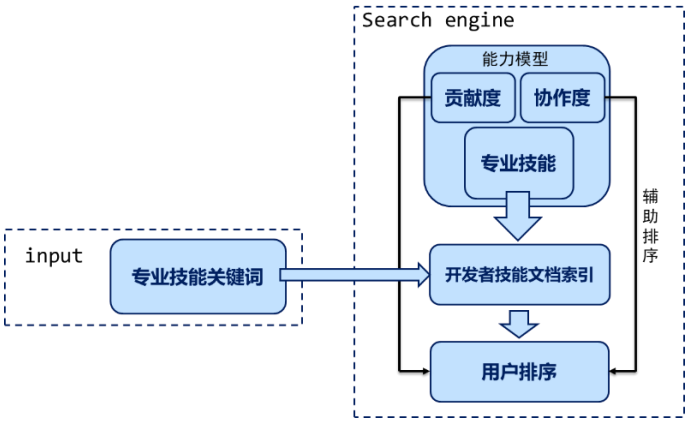


Fig. 8 Capability based developer search

图 8 基于能力的开发者搜索

根据查询关键词以及技能文档索引,搜索出候选开发者集合.为了提高搜索结果的准确性,本文用开发者能力模型的三个维度即专业技能得分、贡献度、协作度来对候选开发者集合排序.对开发者计算其总分的过程与评估开发者一致,不同的是这里每个维度的权重可以设置为开发者在总体候选集合中的排名百分比,并且需要

对贡献度和协作度进行归一化处理.

搜索的另外一个内容就是对开发资源的搜索.由于各个社区已经有相关的搜索功能,因此本文提供开发资源搜索的关注点在于以资源为依托的开发者协作关系可视化.

第三部分是推荐算法,指的是开发者/开发资源的推荐算法.关于这部分内容已经在第 4 节中介绍,因此不再详述.

5.4 服务层

服务层是本系统总的功能的集成,它是对算法层检索到的知识的智能释放的过程.从框架图中可以看到,本系统提供的服务包括六个部分,即数据统计分析、开发者搜索、开发资源搜索、协作关系分析、开发者推荐以及开发资源推荐服务.

(1) 数据统计分析.为了让用户了解本系统的数据类型、数据总量以及其相关规律,本文对获取到的原始数据进行了统计分析与可视化展示.其中,统计主要面向每个社区的开发者与开发资源.

(2) 开发者搜索.在开发者搜索算法的基础上,本文提供了按照基本信息和按照技能关键词的两种搜索方式,重点是开发者的技能搜索. 此外,也可以通过用户名等字段精确搜索开发者.按照第 3 节中介绍的开发者能力建模的方法,对搜索出的开发者进行能力分析,并通过多种形式展示开发者的整体技能.

(3) 开发资源搜索.对于给定的资源关键词,本系统提供了相关的资源搜索服务.搜索结果为相关的资源的简略信息、对应的原始链接以及相关开发者的协作可视化分析.

(4) 协作关系分析.对于开发者协作关系模型,本系统也同样地做了可视化的展示.按照不同的分析方法,协作关系分析包含四个部分:单个开发者的协作关系分析、资源主导的多个开发者协作关系分析、开发者协作社群分析以及开发者兴趣群体分析.

(5) 开发者推荐.基于开发者推荐算法,本系统提供 web 服务以及工具两种服务形式.举例来说,对于 Topcoder 中的开发者推荐,一方面用户可以输入具体的众包任务描述信息,经过本文的推荐算法计算后给出相匹配的开发者列表,这种推荐方式是一种显示的推荐.另一方面本系统将开发者推荐算法集成到浏览器插件中,当用户在 Topcoder 中提交了任务之后,此插件及时给出推荐的开发者列表,这种推荐方式是一种隐式的推荐.

(6) 开发资源推荐.基于代码、问答等开发资源推荐算法,本系统可提供上文所述的显式与隐式的推荐服务.所不同的是,由于开发资源面向开发过程,因此,为了让用户更方便地获取到需要的资源,工具形式为 Eclipse 等 IDE 插件.例如,当开发者需要查找问答资源时,本系统可提供问答推荐插件根据开发环境上下文及时自动地推荐出相关的问答资源.

5.5 讨论

在实现智能协作的集成环境后,本文对如何最大化的发挥该环境的作用来为开发者提供服务进行探讨,表现在多个方面:

(1) 服务于互联网软件开发社区.本文的研究大多以互联网上获取到的数据为基础,因此将回馈于互联网社区作为最直接的出发点,发挥在开发者社区中的影响力.例如,针对 GitHub 的 Reviewer 推荐,基于本文介绍的算法,研发对应的 Web 浏览器插件,当项目管理者面临新的 Pull Request 时,该插件实时推荐出合适的 Reviewer.

(2) 服务于企业.实际上,本文提出的开发者知识图谱或者推荐算法是一种通用的智能化协作开发方法,利用开源数据做了研究示范.在获取到企业开发者的开发数据后,经过适配性改造本文的方法,能够形成面向多种开发任务的智能服务.

(3) 其他应用.在企业进行招聘时,也面临着评估应聘的开发者能力的问题.将本文提出的开发者能力模型为参考,以其 GitHub 等互联网社区账户为依据,可自动化分析其开发能力.另外,本文的开发者能力模型也为开发者的自我评估提供了重要途径,帮助其更好地了解个人能力以作出针对性的改变;而根据开发者协作关系模型则能够帮助开发者找到合适的协作者.

6 总结与展望

本文从软件开发者的角度对软件智能化开发进行了开辟性的探索,为面向开发者智能协作的研究提供了新思路,同时围绕其中关键技术介绍了已开展的研究工作.在对开发者能力评估模型和开发者协作关系模型进行了全面阐述后,提出了新的开发者协作分析方法,构建了开发者知识图谱.针对开发过程中存在的问题,本文提出了面向开发任务的开发者推荐算法.以这些关键技术为基础,形成了一体化的智能协作开发环境,为开发者高效地进行高质量的软件开发提供了便利的条件.

在未来工作中,一方面,由于本文研究的智能协作开发环境以辅助开发者进行软件开发为主.我们将按照自顶向下的研究方法继续完善智能协作环境,通过深入理解开发者在开发过程中的需求以及掌握这种需求的变化,提供更多的面向多种开发任务的开发者推荐与开发资源推荐服务,进一步改进智能推荐所依赖的开发者知识图谱的开发者能力模型与开发者协作关系模型.另一方面,在本文提出的这种智能协作环境的基础上,结合深度学习等人工智能方法研制更加智能化的开发者服务环境.

References:

[1] Thongtanunam P, Tantithamthavorn C, Kula R G, et al. Who should review my code? A file location-based code-reviewer recommendation approach for Modern Code Review. SANER 2015: 141-150.

[2] Brooks Jr F P. The mythical man-month (anniversary ed.)[J]. 1995.

[3] DeMarco T, Lister T. Peopleware: productive projects and teams[M]. Addison-Wesley, 2013.

[4] Jones C. Programming productivity[M]. McGraw-Hill, Inc., 1985.

[5] Ellison N B. Social network sites: Definition, history, and scholarship [J]. Journal of Computer‐Mediated Communication, 2007, 13(1): 210-230.

[6] Concurrent Versions System, [online] Available: https://en.wikipedia.org/wiki/Concurrent_Versions_System#cite_note-1.

[7] Apache Subversion, [online] Available: https://en.wikipedia.org/wiki/Apache_Subversion

[8] Git, [online] Available: https://en.wikipedia.org/wiki/Git

[9] Bugzilla,[online] Available: https://en.wikipedia.org/wiki/Bugzilla

[10] Hahn J, Moon J Y, Zhang C. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties [J]. Information Systems Research, 2008, 19(3): 369-391.

[11] Abreu R, Premraj R. How developer communication frequency relates to bug introducing changes[C]//Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops. ACM, 2009: 153-158.

[12] Yaxin Liu, Peng He, Gaoyan Wu, Yilu Li.Towards Understanding Developers' Collaborative Behavior in Open Source Software Ecosystems [J]. Journal of Software, 2017, 12(6): 393-405.

[13] Hattori L. Enhancing collaboration of multi-developer projects with synchronous changes[C]//Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2. ACM, 2010: 377-380.

[14] Blincoe K. Timely detection of coordination requirements to support collaboration among software developers[C]//Software Engineering (ICSE), 2012 34th International Conference on. IEEE, 2012: 1601-1603.

[15] Sindhgatta R. Identifying domain expertise of developers from source code[C]//Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008: 981-989.

[16] Matter D, Kuhn A, Nierstrasz O. Assigning bug reports using a vocabulary-based expertise model of developers[C]//Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on. IEEE, 2009: 131-140.

[17] Teyton C, Palyart M, Falleri J R, et al. Automatic extraction of developer expertise[C]//Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2014: 8.

[18] Fritz T, Murphy G C, Murphy-Hill E, et al. Degree-of-knowledge: Modeling a developer's knowledge of code[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2014, 23(2): 14.

[19] Lima J, Treude C, Figueira Filho F, et al. Assessing developer contribution with repository mining-based metrics[C]//Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on. IEEE, 2015: 536-540.

[20] Greene G J, Fischer B. Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions[C]//Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ACM, 2016: 804-809.

[21] Huang W, Mo W, Shen B, et al. CPDScorer: Modeling and Evaluating Developer Programming Ability across Software Communities[C]//SEKE. 2016: 87-92.

[22] Constantinou E, Kapitsaki G M. Identifying Developers' Expertise in Social Coding Platforms[C]//Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on. IEEE, 2016: 63-67.

[23] Liu X, Iyer B. Design architecture, developer networks and performance of open source software projects[J]. ICIS 2007 Proceedings, 2007: 90.

[24] Meneely A, Williams L, Snipes W, et al. Predicting failures with developer networks and social network analysis[C]//Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. ACM, 2008: 13-23.

[25] Wolf T, Schroter A, Damian D, et al. Predicting build failures using social network analysis on developer communication[C]//Proceedings of the 31st International Conference on Software Engineering. IEEE Computer Society, 2009: 1-11.

[26] Jermakovics A, Sillitti A, Succi G. Mining and visualizing developer networks from version control systems[C]//Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering. ACM, 2011: 24-31.

[27] Caglayan B, Bener A B, Miranskyy A. Emergence of developer teams in the collaboration network[C]//Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on. IEEE, 2013: 33-40.

[28] Joblin M, Apel S, Hunsen C, et al. Classifying developers into core and peripheral: An empirical study on count and network metrics[C]//Proceedings of the 39th International Conference on Software Engineering. IEEE Press, 2017: 164-174.

[29] Zanetti M S. The co-evolution of socio-technical structures in sustainable software development: lessons from the open source software communities[C]. international conference on software engineering, 2012: 1587-1590.

[30] Zanetti M S, Scholtes I, Tessone C J, et al. Categorizing bugs with social networks: a case study on four open source software communities[C]. international conference on software engineering, 2013: 1032-1041.

[31] Garcia D, Zanetti M S, Schweitzer F, et al. The Role of Emotions in Contributors Activity: A Case Study on the GENTOO Community[C]. 2013: 410-417.

[32] Scholtes I, Mavrodiev P, Schweitzer F, et al. From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects[J]. Empirical Software Engineering, 2016, 21(2): 642-683.

[33] Wu W, Zhang W, Yang Y, et al. Drex: Developer recommendation with k-nearest-neighbor search and expertise ranking[C]//Software Engineering Conference (APSEC), 2011 18th Asia Pacific. IEEE, 2011: 389-396.

[34] Zhu J, Shen B, Hu F. A learning to rank framework for developer recommendation in software crowdsourcing[C]//Software Engineering Conference (APSEC), 2015 Asia-Pacific. IEEE, 2015: 285-292.

[35] Zhang X, Wang T, Yin G, et al. DevRec: A Developer Recommendation System for Open Source Repositories[C]//International Conference on Software Reuse. Springer, Cham, 2017: 3-11.

[36] Hannebauer C, Patalas M, Stünkelt S, et al. Automatically recommending code reviewers based on their expertise: An empirical comparison[C]//Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference on. IEEE, 2016: 99-110.

[37] Holmes R, Murphy G C. Using structural context to recommend source code examples[C]//Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on. IEEE, 2005: 117-125.

[38] Nguyen A T, Hilton M, Codoban M, et al. API code recommendation using statistical learning from fine-grained changes[C]//Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016: 511-522.

[39] Chen X, Zheng Z, Liu X, et al. Personalized QoS-Aware Web Service Recommendation and Visualization[J]. IEEE Transactions on Services Computing, 2013, 6(1): 35-47.

[40] Zhang W, Sun H, Liu X, et al. Temporal QoS-aware web service recommendation via non-negative tensor factorization[J]. International World Wide Web Conferences, 2014: 585-596.

[41] Bai Y, Yin G, Wang H. Multi-dimensions of Developer Trustworthiness Assessment in OSS Community[C]//Trust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE, 2014: 75-81.

[42] Lima J, Treude C, Figueira Filho F, et al. Assessing developer contribution with repository mining-based metrics[C]//Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on. IEEE, 2015: 536-540.

[43] Tangari G C, Maia M A. Ranking Developers' Importance Factors Based on Team Leader Perspective[J]. International Journal of Software Engineering and Knowledge Engineering, 2015, 25(09n10): 1693-1698.

[44] Zizhe Wang, Hailong Sun, Yang Fu, et al. Recommending Crowdsourced Software Developers in Consideration of Skill Improvement[C]//The 32nd IEEE/ACM International Conference on Automated Software Engineering. 2017

[45] Yang Fu, Hailong Sun, Luting Ye. Competition-Aware Task Routing for Contest Based Crowdsourced Software Development[C]//Proceedings of the 6th International Workshop on Software Mining. 2017

[46] Zhenglin Xia, Hailong Sun, Jing Jiang, Xu Wang, Xudong Liu. A Hybrid Approach to Code Reviewer Recommendation with Collaborative Filtering [C]//Proceedings of the 6th International Workshop on Software Mining. 2017

附中文参考文献:

[2] Brooks F P, Li Q. 人月神话[M]. 人民邮电出版社, 2007.