数据驱动的软件开发者智能协作技术

张建[1,2,3]，孟祥鑫[1,2,3]，孙海龙[1,2,3]，王旭[1,2,3]，刘旭东[1,2,3]
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. 北京航空航天大学大数据科学与脑机智能高精尖创新中心，北京 100191；
3. 北京航空航天大学计算机学院，北京 100191

摘要
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。针对此问题，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。首先，收集并分析了海量的开发者相关数据；第二，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；第三，以开发者知识图谱为支撑，阐述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系统；最后，对未来的工作进行了展望。

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

中图分类号：TP311.5 文献标识码：A doi: 10.11959/j.issn.2096-0271.2021006

Data driven intelligent collaboration of software developers

ZHANG Jian[1, 2, 3], MENG Xiangxin[1, 2, 3], SUN Hailong[1, 2, 3], WANG Xu[1, 2, 3], LIU Xudong[1, 2, 3]
1. State Key Laboratory for Software Development Environment (Beihang University), Beijing 100191, China
2. Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China
3. School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Abstract
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, the key technologies for intelligent collaboration through in-depth analysis of developer behavior were studied. Besides, the corresponding support environment was also developed on the basis of

the key technologies to improve the efficiency and quality of software development. Firstly, a large amount of data related to developers were collected and analyzed. Secondly, a systematic approach of analyzing developers and their collaboration which is called developer knowledge graph was proposed. Thirdly, supported by the developer knowledge graph, the collaborative development method based on intelligent recommendation was introduced thoroughly. Depending on the above technologies, the corresponding supporting tools were developed, and a system of intelligent collaborative development environment was provided. Finally, the future work was prospected.

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

1 引言

软件开发是以“开发者”为中心的智力密集型活动。尽管在模型驱动和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，但是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开发任务难以由单个开发人员独立完成，通常需要开发者群体的共同协作。例如，据报道，Windows 7的开发涉及2 000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的开发者数量庞大，且开发者相互之间了解有限，因此，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5 600多万个注册用户，其中拉取请求（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，一个拉取请求得到用户评论的平均时间会延长12天[1]；软件开发问答社区Stack Overflow汇聚了1 300多万个用户，但仍然有超过620万个问题（约占总问题的30%）未得到有效回答，其主要原因是缺乏针对问题进行回答者推荐的支持。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《人月神话》一书中明确指出：简单地基于“人月”来度量和计划软件开发任务存在不足[2]。例如，软件开发成本会随着开发人数和时间的变化而发生变化，但进度却可能变缓甚至停滞。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、交互和作用机理，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4个方面：直接的沟通与交流、开发任务的分配、开发结果的汇聚、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开发者的主观经验进行，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，交流的对象往往局限于开发团队内相互熟悉的人员，但是团队外或者互联网上可能存在大量潜在的协作者；在

开发任务的分配上，一般由技术负责人依据其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积累的软件资源库，而缺乏对互联网上更大范围的资源的重用支持。因此，基于互联网中的海量开发者相关数据，近年来大量的工作研究了面向特定开发任务的开发者推荐和开发资源推荐。一方面，开发者推荐面向开发过程中的不同阶段，推荐合适的开发人员，包括代码评审者推荐[3-4]、开发者以及开发团队推荐[5-7]等；另一方面，开发资源推荐推荐的是开发过程中需要的软件资源，包括应用程序接口（application programming interface，API）推荐[8]、代码推荐[9-10]和代码修复补丁推荐[11]等。然而，这类工作只关注与特定任务相关的历史数据，忽略了对开发者及其协作关系的分析，使得推荐性能受到限制。此外，目前还没有形成系统化的软件开发者智能协作环境，不能很好地支撑高效的软件协作开发。

针对软件开发中开发者协作面临的问题，本文以提高开发者的协作效率为目标，研究智能化的群体协作方法，并开发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积累的软件开发大数据进行分析，对开发者的能力特征和历史协作行为等进行定性和定量的分析，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，基于开发者关联关系进行分析，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。

总体来说，本文的主要贡献如下：

- 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；
- 提出了开发者能力模型与协作关系模型，并基于从互联网上收集的软件开发大数据构建了开发者知识图谱；
- 在开发者知识图谱的基础上，针对开发者的智能推荐和开发资源推荐，提出了若干关键技术，并基于真实数据进行了实验评估；
- 研制了一个智能协作开发支撑环境的系统，并开展了应用实践。

2 智能协作框架

开发者协作一直是软件开发中的关键问题。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协作[12]，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[13]。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诸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，尤其是在社交媒体与社交网络（social network）[14]被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[15-19]。

本文在现有研究的基础上，提出了智能协作支撑环境的框架：收集软件开发者协作开发产生的多源异构大数据，提取出其中蕴含的软件开发知识，并以知识图谱的形式进行组织，进一步借助于搜索、推荐等技术，将这些知识应用于软件开发活动，从而实现软件开发者的智能协作，最终为提高软件开发效率与质量提供支撑。整体的研究框架如**图1**所示。

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，围绕开发者知识的提取与利用，需要解决4类关键问题。

- 数据源。从数据源增量爬取的数据是多源异构的，这种原始数据包含杂乱无章的信息。这些信息对于推荐或搜索层来说价值很低，不能被直接使用，需要对这些数据进行有效管理和知识提取。

- 开发者知识库。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：开发者能力评估技术和开发者关系挖掘技术。开发者能力评估技术的核心是建立定性定量相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。

- 推荐与搜索。构建知识库的目标是将其中的知识应用到软件开发过程中，具体的方式包括面向智能协作的知识搜索以及知识推荐。其中，搜索技术既包括开发者及其关联资源的搜索，也包括开发者的能力以及开发者之间的关系搜索；智能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。

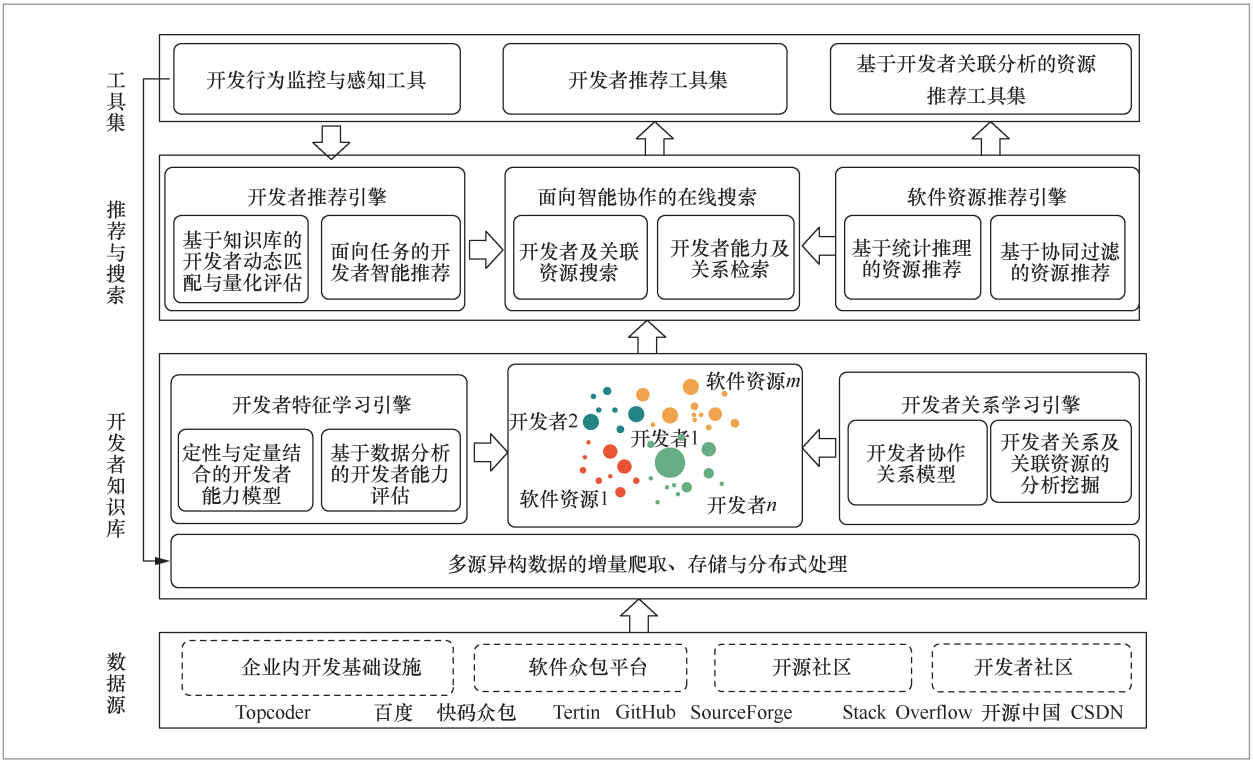


图1 面向软件开发的智能化群体协作框架

● 工具集。其主要包括开发者推荐和开发资源推荐工具，同时，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，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4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→开发者知识图谱的更新。

因此，智能协作的核心是对开发者知识的利用。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）。该方法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

开发者知识图谱是将开发者作为知识结点、将开发者协作关系作为边的知识图谱。其中，结点描述了开发者的能力特征，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。在本文的工作中，为了覆盖更多的开发知识，笔者也将开发资源结点以及开发者与开发资源的关系纳入了知识图谱。

3.1 开发者能力模型

现有的开发者能力评估方法存在一些不足[20-22]：第一，大多数的能力评估方法只注重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，现有方法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，也体现在不同的开发活动（如问答、测试等）中。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是需要解决的一个难题。

针对以上问题，本文提出了一种新的开发者能力模型，从多个维度度量开发者的能力，如图2所示。

本文刻画了开发者能力的3个主要方面，并且基于度量的灵活性，建立了开发者能力树。具体来说，树的根结点为开发者能力，其子树分别为专业技能（skill）、贡献度（contribution）以及协作度（collaboration）。本文对能力的刻画以统计方法为基础，该方法更适用于大型社区（如GitHub、Stack Overflow），且可解释性较强。下面分别对这3个维度进行具体说明。

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}集合的形式。在该集合中，key为Java的技能的评分为6，在某种评分体系下，表示开发者能够较熟练地掌握Java语言，其他技能依此类推。关于技能的抽取和评分，可以集成已有的研究方法。本文针对活跃在不同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，则开发者在该技能标签T下的技能得分为$S_T=\sum_{\alpha_i\in A}s_i$。类似

地，在其他社区中也主要采用基于统计分析的方式对开发者技能进行评估。

贡献度是指开发者在特定社区中对该社区作出的贡献，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。这些贡献指的是开发者的各项活动，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开发者的贡献，笔者首先抽取开发者不同贡献的集合$C=\{C_i|i=1,2,3,\cdots,N\}$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。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（follow），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。以某个开发者的专业技能为例，假设其中包含N个技能标签$T_1,T_2,\cdots,T_N$，对应的得分为$S_1,S_2,\cdots,S_N$，按照某种任务需求，其技能权重为$\omega_1,\omega_2,\cdots,\omega_N$，则其专业技能的整体得分为：

$$P_1=\sum_{i=1}^{N}\omega_i S_i \qquad (1)$$

同样，可得到贡献度和协作度维度下的得分，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：

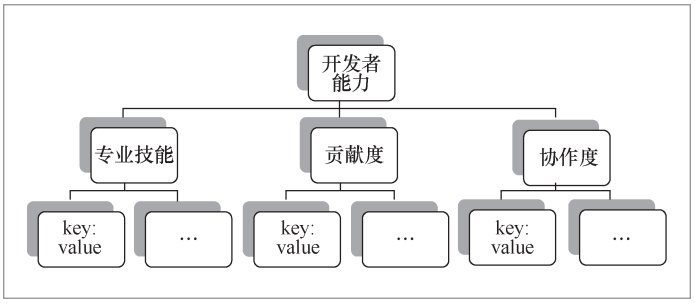


图2 软件开发者能力模型

$$S=\sum_{j=1}^{3}\mu_j P_j \qquad (2)$$

这项得分给出了开发者的整体能力。

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，因为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，现有方法也未对协作关系强度进行度量。

本文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。其在知识图谱中被表示为开发者结点之间的边，这些边包含的协作关系如**图3**所示。

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，将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

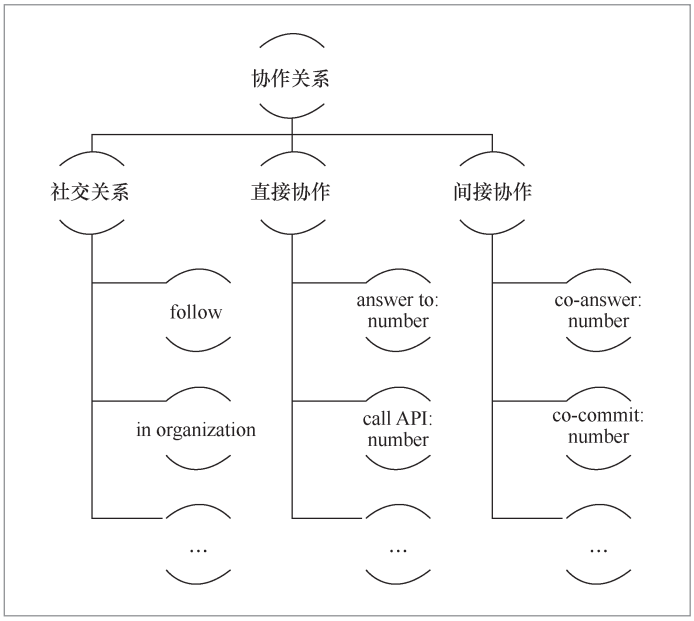


图 3 软件开发者协作关系模型

社区中的人际关系。虽然与开发活动并不直接相关，但是社交关系会对软件开发效率产生重要影响。例如，社交关系又可以分为follow关系和in organization关系等，分别表示开发者之间的互相关注和开发者隶属于同一个组织等。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、D_1实现某些功能时调用了D_2提供的接口，number表示这些协作的频度。此外，开发者共同修改代码文件、对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。间接协作指开发者之间的间接交互关系。相对于直接协作来说，它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。

此外，笔者还提出了众包社区中开发者能力演化模型[23]、基于知识追踪的众包软件开发者能力评估方法[24]、基于程序代码分析的开发者能力刻画方法[25]、实体级的开发者情感分析方法[26]和跨社区的软件开发者画像方法[27]等。

4 基于开发者知识图谱的智能推荐方法

在大规模软件开发历史数据的基础上，借助对开发者进行建模分析形成的开发者知识图谱，可提供开发者推荐与开发资源推荐两种智能服务。

（1）开发者推荐

解决当前软件团队组织的相对固化、依靠主观经验、准确性不高问题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，开发活动之间存在关联依赖，所以开发者的智能推荐具有个性化需求、复杂度高的特点。本文将介绍面向编码、测试以及软件开发问答等任务的开发者智能推荐和任务分配方法，实现对软件开发任务需求与开发者的智能匹配。基于开发者知识图谱中的开发者能力建模与协作关系分析，结合传统机器学习以及深度学习方法，笔者研究了开发者[28-31]、开发团队[32]的智能推荐。例如，针对软件问答社区中越来越多的问题无法得到及时回答导致的开发效率下降的情况，笔者提出了一种新的利用文本潜在隐含主题的回答者推荐方法[28]，为了提高推荐的可信度和准确率，利用开发者之间的协作关系优化推荐

效果，并以此为核心设计了一种有较高准确率和可用性的面向软件开发的回答者推荐系统。这里仅以众包软件开发为代表性案例（见第4.1节和第4.2节），详细阐述其中的开发者以及开发团队智能推荐技术。

（2）开发资源推荐

重用软件开发社区中积累的开发资源（如代码、问答知识等）是提高软件开发效率的重要途径，但这些资源语义结构复杂、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，给开发者获取开发资源带来了很多困难。笔者通过检索开发者知识图谱中与开发者关联的海量软件资源数据，提出了对这些结构复杂的数据进行语义解析的方法，为开发者推荐在开发过程中所需要的资源，从而实现以开发资源为中心的隐式协作开发。具体来说，笔者研究了上下文感知的编程问答资源推荐[33]、面向高效编程的API使用模式推荐[34]、软件问答社区中的Tag推荐[35]等技术。本文以面向高效编程的API使用模式推荐为例（见第4.3节），阐述开发者资源智能推荐技术。

4.1 面向众包软件开发的开发团队推荐技术

近年来，基于互联网的众包软件开发成为一种新型的软件开发模式，为管理者提供了发布需求的平台，也为开发者提供了自由选择开发任务的机会。然而，相比于数据标注等传统任务，软件开发更加复杂，因此往往需要团队协作。现有的团队推荐算法仅适用于管理者将任务指派给特定开发团队的场景，不适用于众包软件开发任务场景。本文提出了一种基于开发者队友选择偏好的团队推荐算法[32]，根据软件开发任务的技能需求与人员需求，考虑各开发者的队友选择偏好，为开发者推荐高协作意愿团队，以确保团队的组建成功率。

本文的团队推荐算法首先对开发者进行能力建模和亲密度建模，然后对协作意愿影响因素建模，分析开发者队友选择偏好，基于开发者的队友选择偏好建模团队协作意愿，设计一种基于贪心策略的近似算法，为开发者推荐高协作意愿的团队，以确保团队组建成功率，且适用于大规模众包软件开发平台。本文个性化团队推荐算法框架如**图4**所示，主要由以下3个模块构成。

- 数据模型：该模块对开发者进行能力建模和亲密度建模。
- 团队协作意愿建模：该模块首先给出协作意愿影响因素的数学表示，然后分析开发者的队友选择偏好，基于开发者的队友选择偏好建模团队协作意愿。
- 近似算法：该近似算法基于贪心策略为开发者推荐高协作意愿的团队，适用于大规模众包软件开发平台。

实验结果表明，本文算法推荐结果明显优于亲密度优先（closeness first，

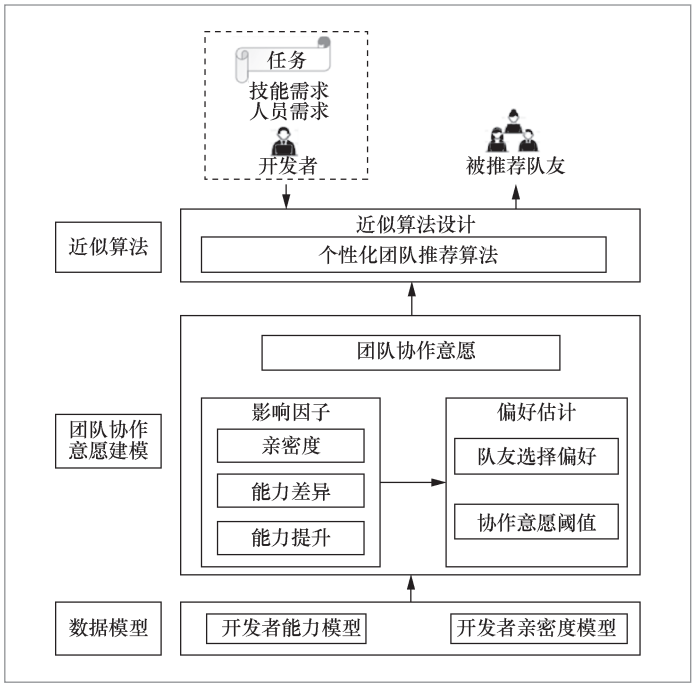


图 4 个性化团队推荐算法框架

CF）、能力提升优先（expertise gain first，EGF）、能力差异优先（expertise difference first，EDF）3个对比方法，本文算法推荐出的开发团队存在更高的团队协作意愿，保证了开发者的团队组建成功率。本文同时考虑了影响团队协作意愿的3个因素，而CF、EGF、EDF方法仅考虑了其中一个因素。Optimal方法的推荐结果是理论最优解，该方法遍历开发者集合以找到推荐的理论最优解，但其算法时间复杂度为$O(n^N)$（其中，n表示众包开发者集合大小，N表示团队规模），复杂度高，当数据规模很大时，无法达到实时推荐的效果，不适用于大规模众包软件开发平台。本文提出的近似算法在保证推荐有效性的同时，算法的时间复杂度为$O(n\cdot N)$（其中，n表示众包开发者集合大小，N表示团队规模），确保了推荐的效率。

4.2 基于元学习的众包软件开发者推荐方法

尽管众包软件开发模式日益发展，但仍然存在一个典型的问题，即开发者与任务的匹配问题。对于具有严格的时间要求的软件开发任务，如何借助历史数据评估开发者的能力并依此推荐可靠的开发者，以保障开发效率和质量，成为研究的关键点。现在的众包软件开发者推荐方法仅能针对大数据集进行分析建模，缺乏对小数据集的研究。多标签分类需要大量的数据才能训练出一个非常优秀的模型，这限制了该方法在数据量不足时的应用。此外，基于用户角色的数据角色预测方法需要使用用户的历史数据，小数据集上的用户历史显然太少，这导致难以训练模型。现有技术也缺乏针对冷启动问题的解决方法。在多标签形式的分类问题中，其基本假设为只有注册次数高于一定阈值时才是可靠的，那些获胜次数少的用户会被过滤，导致有些用户即使有能力完成任务也难以被算法推荐出来。而基于用户角色的预测依旧需要大量的用户历史数据才能训练分类器，对于那些缺乏相应历史的用户，则无法处理。

本研究针对众包软件开发者推荐的问题，提出了一种基于元学习的众包软件开发者推荐方法[31]，从而保证软件质量和开发效率。如**图5**所示，进行推荐时，所述方法中包含3个主要组件：用户注册行为预测器、用户提交行为预测器、用户获胜行为预测器。这三者分别可以预测用户的相应行为，用户提交行为预测器基于用户注册后的情况进行预测，而用户获胜行为预测器基于用户提交后的情况进行预测，用户注册行为预测器的预测没有任何先决条件。依据用户获胜行为预测器预测出的获胜概率p，用户注册行为预测器和用户提交行为预测器会相继对预测出的注册概率和提交概率不在前Top_R、Top_S的用户的获胜概率进行修改，R、S为0到1的阈值，可以人为设定。如果把获胜概率p设为0，此时直接结束模型对该输入的预测行为。对于这3个组件，采用元模型进行构建，基础学习模型使用Extrees、XGBoost、深度神经网络（deep neural network，DNN）3个算法，并依据原学习方法进行分析算法选择。

实验结果（见**表1**）表明，相比于协同过滤分类（collaborative filtering classification，CBC）、基于内容的推荐（CrowdRex）、动态众包工人决策支持（dynamic crowd worker decision support，DCW-DS）的方法，本文提出的元学习模型（policy model）能够在众包软件开发平台Topcoder上获得当前较高的推荐准确率。其中，Acc@3表示Top-3的推荐准确率，依此类推。MRR是国际上通用

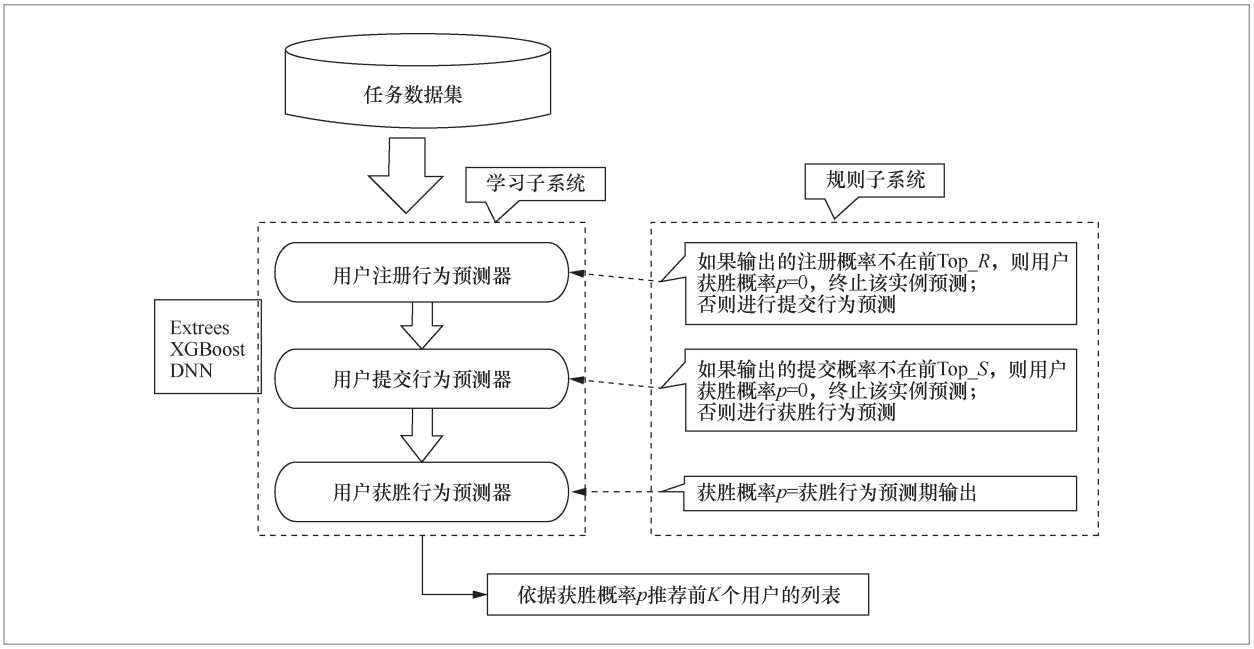


图 5 众包开发者推荐的整体流程

的对搜索/推荐算法进行评价的机制，即如果第一个结果即可匹配，则分数为1；第二个结果匹配，则分数为0.5；第n个结果匹配，则分数为$1/n$；如果没有匹配的结果，则分数为0。单个方法在该数据集上的最终MRR分数为所有匹配分数的平均值。

表1 研究方案在 Topcoder 数据集上的性能对比

方法	Acc@3	ACC@5	ACC@10	MRR
CBC	0.143	0.170	0.228	0.161
CrowdRex	0.081	0.094	0.175	0.119
DCW-DS	0.060	0.093	0.178	0.088
本文方法	0.467	0.571	0.581	0.312

4.3 面向高效编程的API使用模式推荐技术

开发者在开发过程中往往需要寻求其他开发者的帮助。开发者或者仅与熟悉的开发者沟通，或者通过互联网提出问题，两者的效率和质量均得不到保障。为了提高隐式协作的效率，本文研究了API使用模式的智能推荐方法，即给定开发者需求，推荐其他开发者产生的相应的API使用模式代码。在根据自然语言自动生成代码的问题上，目前有工作尝试生成完整代码，或者生成API序列。前者生成代码的质量往往比较低，很难达到直接可用的效果；而后者难以直接被用于补全源代码。相比之下，包含API及其相关控制流语句的结构（称之为API使用模式）有适中的复杂度，并且可以提供足够的代码骨架信息来帮助开发者实现各种各样的功能。根据自然语言查询生成相关的API使用模式，既可以达到比较好的生成效果，又可以有效地帮助开发者。

本文针对由自然语言查询生成API使用模式的问题，提出了一种新的API使用模式推荐技术[34]。本研究采用编码器-解码器模型，设计了API-MCTree解码器模块，并对该模型设计了训练和预测的算法。为了获得<自然语言查询，API使用模式>形式的数据对作为训练数据，还需要获得对应API使用模式的自然语言查询。根据

Javadoc说明指导，一个函数的注释文档的第一句可以作为当前函数的简短描述，因此抽取其中的第一句作为自然语言查询。为了在学习过程中更好地表示API使用模式，每个API使用模式可以被看成一个有约束的树形结构。假设每种控制流结构都有一种不同的颜色，称这种有约束的树为API-MCTree（API multi-colored tree）。本研究基于一种带注意力机制的编码器-解码器模型。编码器部分是基于循环神经网络模型的，具体使用的是长短期记忆（LSTM）单元，它可以以循环的方式一个接一个地处理自然语言的词输入，然后将自然语言序列编码到一个固定大小的向量。解码器根据自然语言向量$\boldsymbol{c}$生成API-MCTree。对于API使用模式这种特殊的形式，本文设计了一种新型的解码器，称之为API-MCTree解码器，如**图6**所示。与序列解码器不同，在时间步t，API-MCTree解码器的隐藏状态不仅依赖于上一个时间步t-1，也依赖于父亲非终结符结点的信息。因而，在模型中采用了parent-feeding连接机制。在时间步t，将父亲非终结符结点的隐藏状态和输入向量拼接在一起，然后把它们共同送入LSTM单元，基于输出的隐藏状态生成相应的树形API使用模式。

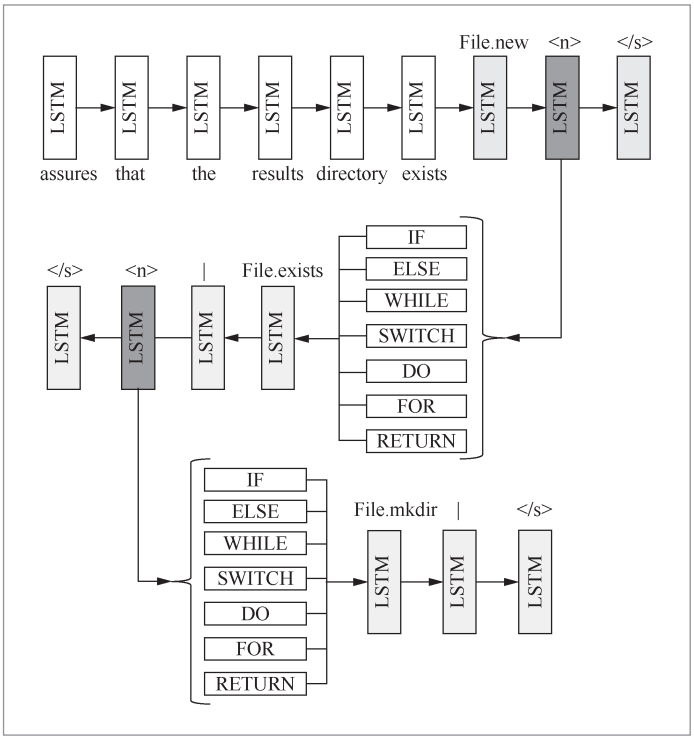


图 6 API-MCTree 解码器

为了验证本文设计的API使用模式生成方法的有效性，分析模型中设计的合理性并与其他相关方法进行对比，笔者设计了相关的实验进行评估。与其他相关工作的评价方式类似，使用BLEU-4来衡量生成的API使用模式的质量，它可以量化候选序列和真实序列之间的相似度。笔者比较了本文方法和现有的4种方法。**表2**是这些方法在具体指标上的表现，可以看到，本文提出的方法拥有最高的BLEU分数（44.90%）和最高的测试准确度（37.45%）。通过和对比方法的比较可知，本文提出的方法有较好的效果，在自然语言查询语义理解、API使用模式的语法和语义信息的捕捉和表达上具有不错的能力。

5 大规模开发者智能协作支撑环境

如**图7**所示，基于对开发者画像、开发者推荐及开发资源推荐等关键技术的研究，笔者构建了大规模开发者智能协作支撑环境，自下而上可分为5个层次。

（1）数据源和数据管理

开发了面向GitHub、Stack Overflow、Topcoder、CSDN、GitLab平台的分布式爬虫工具，获取了源代码、代码注释、项目文档、开发者行为、技术问答、技术博客等海量开发数据，并存储到MySQL、MongoDB、Neo4j数据库中（见**表3**）。

（2）开发者知识库

定义了开发者知识库的基本模型，从原始数据中提取开发结点与开发关系，以

构建开发者知识库。开发结点包括开发者以及开发者的技能、效率、情感等信息，还包括开发资源及其主题、类别、质量等信息；开发关系包含开发者间的社交、竞争、协作等关系信息。最终，建立起一个包含24亿个结点、80.7亿条边的大规模开发者知识库，其中包含4 397万名开发者，数据总量达到13 TB（见**表3**）。

（3）智能推荐与搜索

根据已构建的开发者知识库，分别进行了开发者推荐、开发资源推荐、知识库搜索3个方面的研究。对于开发者推荐，使用协同过滤、元学习模型、主题模型、协作关系感知等方法，对开发过程中不同阶段

表 2　本文方法与现有方法的性能比较

方法	BLEU分数	测试准确度
Lucene+Clustering	19.78%	13.17%
SWIM	21.62%	18.64%
Seq2Seq	34.37%	30.67%
Seq2Tree	37.81%	34.17%
本文方法	44.90%	37.45%

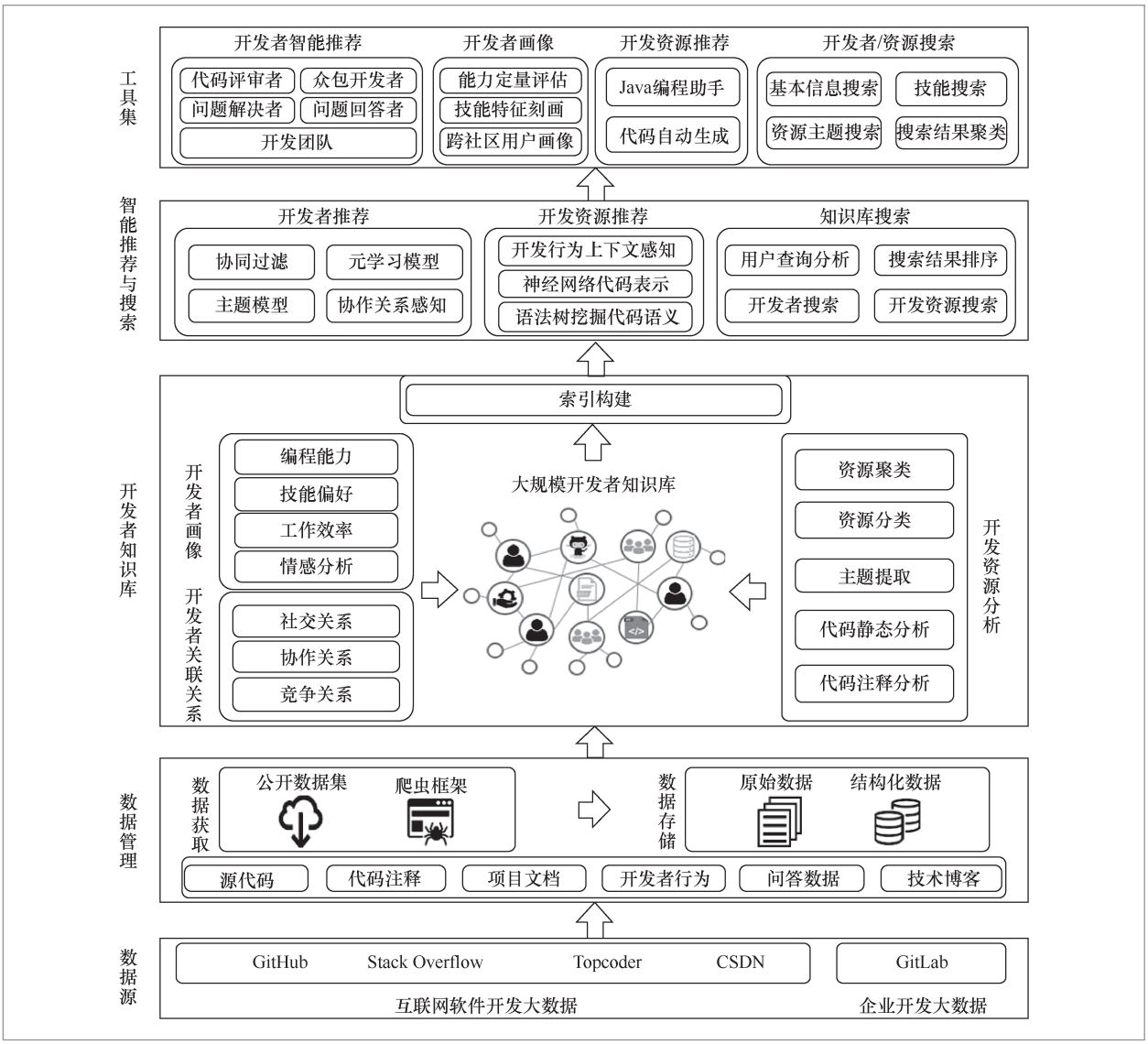


图 7　大规模软件开发者智能协作支撑环境体系结构

表3 系统中汇聚的主要软件大数据情况

项目		GitHub	Stack Overflow	Topcoder	CSDN	企业数据	
						东软集团股份有限公司	万达信息股份有限公司
数据类型		开发过程数据	问答数据	众包开发数据	博客、论坛、问答	开发过程数据GitLab	
数据采集		API+直接下载	直接下载	API	爬虫	实时采集API	实时采集API
数据存储方式		MySQL Neo4j MongoDB	MySQL Neo4j MongoDB	MySQL Neo4j MongoDB	MySQL Neo4j MongoDB	MySQL Neo4j MongoDB	MySQL Neo4j MongoDB
数据量	原始数据	12.9 TB	323 GB	8.1 GB	7.6 GB	N/A	N/A
	资源数量	开发者：3 241万 项目数：1.25亿 commit：13.7亿	用户数：1 053万 问题数：1 774万 回答数：2 711万 标签数：5.50万	开发者：4.05万 项目数：0.39万 提交数：8.69万 挑战数：3.73万	用户：98.7万 博客：104.5万 问答：15.7万 论坛：57.9万	代码提交：3万 行为数据：2亿+ 项目：280+ 开发者：130+ 问题：1.5万	开发者：296 项目：978
	知识图谱	结点数：22.4亿 关系数：74.5亿	结点数：1.6亿 关系数：5.9亿	结点数：7.8万 关系数：1 829万	结点数：706万 关系数：1 294万	N/A	结点数：1.2万 关系：3.7万
加工数据种类		开发者能力属性、开发者协作关系、开发资源					

任务的开发者分配模式进行了优化；对于开发资源推荐，一方面通过开发行为上下文感知方法来捕获开发者的行为特征，另一方面通过神经网络代码表示与语法树挖掘代码语义等方法对相关代码进行结构化表示，最后综合两者作为输入特征，基于知识库中的资源与历史协作关系进行资源推荐；对于知识库搜索，对开发者与开发资源潜在主题的提取技术进行了研究，并辅以多种维度的排序返回方式，提高了搜索的精度。

(4)工具集

基于以上智能推荐与搜索方法，开发了一系列智能开发工具集，包括开发者智能推荐工具集、开发者画像工具集、开发资源推荐工具集、开发者搜索工具集和开发资源搜索工具集。

- 开发者智能推荐工具集包括代码评审者、众包开发者、问题(issue)解决者、问题回答者的推荐工具，这些工具能够在开发过程中的不同阶段优化开发者的分配策略，以提高协作开发的效率。
- 开发者画像工具集包含能力定量评估工具、技能特征刻画工具与跨社区画像集成工具，能够从社区贡献、协作能力、技能偏好、代码质量、开发者情感等维度对开发者能力做定性与定量分析，并且可以对同时使用多个开发平台的开发者进行跨平台开发能力的综合展示。
- 开发资源推荐工具集包括Java编程助手工具与代码自动生成工具，前者能够根据开发者的开发行为特征推荐与代码上下文相关的优质问答资源，后者能够根据自然语言的需求描述，面向Java的JDK库提供API调用序列的自动生成服务，帮助进行简单的函数功能设计。
- 开发者搜索工具集提供了基于基本信息与技能信息的两种搜索模式，支持跨开发平台搜索。
- 开发资源搜索工具集提供了基于资源主题的搜索模式，并可以对搜索结果依据技能标签进行聚类，帮助开发者更好地获取所需的开发资源。

最终，笔者将上述工具集成在大规模开发者智能协作支撑环境iCoOper平台上，并且提供REST API服务，图8～图10

图 8　开发者能力画像

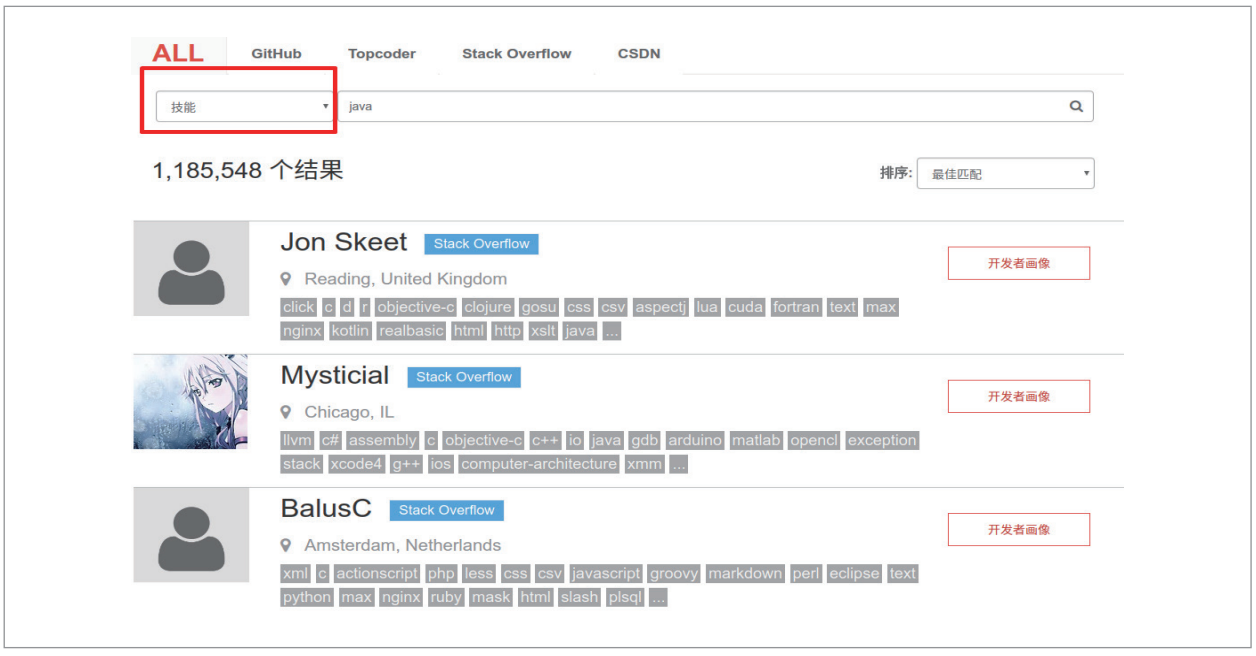


图 9　开发者智能搜索

图 10 代码评审者智能推荐

给出了该平台的部分功能截图。不同企业的开发管理平台中的历史开发数据具有结构一致性，因此iCoOper平台具有良好的可扩展性，能够容易地嵌入更多企业的开发环境中。系统已在东软集团股份有限公司和万达信息股份有限公司等进行部署，系统直接与企业内部代码管理系统GitLab进行对接，自动获取和分析企业内部开发人员的相关数据，构建开发者知识图谱，并实现了对开发者的画像、搜索和推荐等支持，在智慧城市、互联网金融等应用项目的开发中进行了实际应用。

6 结束语

本文从软件开发者的角度对软件智能化开发进行了探索，介绍了开发者智能协作研究的思路和所开展的关键技术研究工作。首先，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。进一步，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以及开发资源智能推荐算法。以这些关键技术为基础，构建了一体化的智能协作开发环境，以期为提高软件开发效率和质量提供有效支撑。

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，笔者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，在本文提出的智能协作环境的基础上，结合深度学习等方法研究并构建更加智能化的开发者服务环境。

参考文献:

[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[C]//2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering. Piscataway: IEEE Press, 2015: 141-150.

[2] BROOKSF P. The mythical man-month (anniversary ed.)[M]. Boston: Addison-Wesley Longman Publishing Co., Inc., 1995.

[3] RAHMAN M M, ROY C K, REDL J, et al. CORRECT: code reviewer recommendation at GitHub for Vendasta technologies[C]//The 31st IEEE/ACM International Conference on Automated Software Engineering. Piscataway: IEEE Press, 2016: 792-797.

[4] ASTHANA S, KUMAR R, BHAGWAN R, et al. WhoDo: automating reviewer suggestions at scale[C]//The 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM Press, 2019: 937-945.

[5] LIU H B, QIAO M, GREENIA D, et al. A machine learning approach to combining

individual strength and team features for team recommendation[C]//2014 13th International Conference on Machine Learning and Applications. Piscataway: IEEE Press, 2014: 213–218.

[6] SAPIENZA A, GOYAL P, FERRARA E. Deep neural networks for optimal team composition[J]. Frontiers in Big Data, 2019, 2: 14.

[7] GAO D W, TONG Y X, SHE J Y, et al. Top-k team recommendation and its variants in spatial crowdsourcing[J]. Data Science and Engineering, 2017, 2(2): 136–150.

[8] NGUYEN A T, HILTON M, CODOBAN M, et al. API code recommendation using statistical learning from fine-grained changes[C]//The 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York: ACM Press, 2016: 511–522.

[9] LUAN S F, YANG D, BARNABY C, et al. Aroma: code recommendation via structural code search[J]. Proceedings of the ACM on Programming Languages, 2019, 3(OOPSLA): 1–28.

[10] SVYATKOVSKIY A, ZHAO Y, FU S Y, et al. Pythia: ai-assisted code completion system[C]//The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM Press, 2019: 2727–2735.

[11] ZHANG X D, ZHU C G, LI Y, et al. Precfix: large-scale patch recommendation by mining defect-patch pairs[C]//The ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice. New York: ACM Press, 2020: 41–50.

[12] DEMARCO T, LISTER T. Peopleware: productive projects and teams[M]. New Jersey: Addison-Wesley, 2013.

[13] JONES C. Programming productivity[M]. New York: McGraw-Hill, Inc., 1985.

[14] BOYD D M, ELLISON N B. Social network sites: definition, history, and scholarship[J]. Journal of Computer-Mediated Communication, 2007, 13(1): 210–230.

[15] MENEELY A, WILLIAMS L, SNIPES W, et al. Predicting failures with developer networks and social network analysis[C]//The 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York: ACM Press, 2008: 13–23.

[16] WOLF T, SCHROTER A, DAMIAN D, et al. Predicting build failures using social network analysis on developer communication[C]//The 31st International Conference on Software Engineering. Piscataway: IEEE Press, 2009: 1–11.

[17] JERMAKOVICS A, SILLITTI A, SUCCI G. Mining and visualizing developer networks from version control systems[C]//The 4th International Workshop on Cooperative and Human Aspects of Software Engineering. New York: ACM Press, 2011: 24–31.

[18] CAGLAYAN B, BENER A B, MIRANSKYY A. Emergence of developer teams in the collaboration network[C]//2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering. Piscataway: IEEE Press, 2013: 33–40.

[19] JOBLIN M, APEL S, HUNSEN C, et al. Classifying developers into core and peripheral: an empirical study on count and network metrics[C]//The 39th International Conference on Software Engineering. Piscataway: IEEE Press, 2017: 164–174.

[20] SINDHGATTA R. Identifying domain expertise of developers from source code[C]//The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2008: 981–989.

[21] MATTER D, KUHN A, NIERSTRASZ O. Assigning bug reports using a vocabulary-based expertise model of developers[C]//The 6th IEEE International Working Conference on Mining Software Repositories.

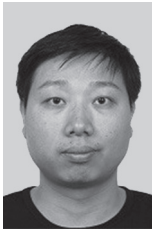
Piscataway: IEEE Press, 2009: 131–140.
[22] TEYTON C, PALYART M, FALLERI J R, et al. Automatic extraction of developer expertise[C]//The 18th International Conference on Evaluation and Assessment in Software Engineering. New York: ACM Press, 2014: 8.
[23] WANG Z Z, SUN H L, FU Y, et al. Recommending crowdsourced software developers in consideration of skill improvement[C]//2017 32nd IEEE/ACM International Conference on Automated Software Engineering. Piscataway: IEEE Press, 2017: 717–722.
[24] WANG Z Z, SUN H L, HAN T. Predicting crowdsourcing worker performance with knowledge tracing[C]//International Conference on Knowledge Science, Engineering and Management. Cham: Springer, 2020: 352–359.
[25] WANG J, MENG X X, WANG H M, et al. An online developer profiling tool based on analysis of GitLab repositories[C]// CCF Conference on Computer Supported Cooperative Work and Social Computing. Singapore: Springer, 2019: 408–417.
[26] DING J, SUN H L, WANG X, et al. Entity-level sentiment analysis of issue comments[C]//The 3rd International Workshop on Emotion Awareness in Software Engineering. New York: ACM Press, 2018: 7–13.
[27] YAN J F, SUN H L, WANG X, et al. Profiling developer expertise across software communities with heterogeneous information network analysis[C]//The 10th Asia-Pacific Symposium on Internetware. New York: ACM Press, 2018: 1–9.
[28] SHAO B, YAN J F. Recommending answerers for stack overflow with LDA model[C]//The 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing. New York: ACM Press, 2017: 80–86.
[29] XIA Z L, SUN H L, JIANG J, et al. A hybrid approach to code reviewer recommendation with collaborative filtering[C]//2017 6th International Workshop on Software Mining. Piscataway: IEEE Press, 2017: 24–31.
[30] FU Y, SUN H L, YE L T. Competition-aware task routing for contest based crowdsourced software development[C]// 2017 6th International Workshop on Software Mining. Piscataway: IEEE Press, 2017: 32–39.
[31] ZHANG Z Y, SUN H L, ZHANG H Y. Developer recommendation for Topcoder through a meta-learning based policy model[J]. Empirical Software Engineering, 2019, 25(1): 1–31.
[32] YE L T, SUN H L, WANG X, et al. Personalized teammate recommendation for crowdsourced software developers[C]//The 33rd ACM/IEEE International Conference on Automated Software Engineering. New York: ACM Press, 2018: 808–813.
[33] SUNF M, WANGX, SUNH L, et al. Recommendflow: use topic model to automatically recommend stack overflow Q&A in IDE[C]//International Conference on Collaborative Computing: Networking, Applications and Worksharing. Cham: Springer, 2016: 521–526.
[34] TIAN Y F, WANG X, SUN H L, et al. Automatically generating API usage patterns from natural language queries[C]//2018 25th Asia-Pacific Software Engineering Conference. Piscataway: IEEE Press, 2018: 59–68.
[35] ZHANG J, SUN H L, TIAN Y F, et al. Poster: semantically enhanced tag recommendation for software CQAs via deep learning[C]//2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion). Piscataway: IEEE Press, 2018: 294–295.

作者简介

张建（1994– ），男，北京航空航天大学计算机学院博士生，主要研究方向为软件工程、源代码分析、自然语言处理。

孟祥鑫（1995– ），男，北京航空航天大学计算机学院博士生，主要研究方向为基于模板的程序自动修复与基于深度学习的程序自动修复。

孙海龙（1979– ），男，博士，北京航空航天大学计算机学院教授、博士生导师，主要研究方向为智能软件工程、群体智能和分布式系统。



王旭（1986– ），男，博士，北京航空航天大学计算机学院讲师，主要研究方向为基于大数据的软件分析和智能化开发。

刘旭东（1965– ），男，博士，北京航空航天大学计算机学院教授、博士生导师，北京航空航天大学计算机学院计算机新技术研究所所长，可信网络计算技术国防重点学科实验室主任，主要研究方向为网络化软件开发方法、可信软件技术、软件中间件技术和信息化标准。

收稿日期： 2020-10-20

通信作者： 孙海龙，sunhl@buaa.edu.cn

基金项目： 国家重点研发计划基金资助项目（No.2016YFB1000800）；国家自然科学基金资助项目（No.61932007，No.61972013）

Foundation Items: The National Key Research and Development Program of China(No.2016YFB1000800), The National Natural Science Foundation of China(No. 61932007, No.61972013)