计算机科学
COMPUTER SCIENCE

开源社区众包任务的开发者推荐方法

蒋竞, 平源, 吴秋迪, 张莉

引用本文

蒋竞, 平源, 吴秋迪, 张莉.开源社区众包任务的开发者推荐方法[J]. 计算机科学, 2022, 49(12): 99-108.

JIANG Jing, PING Yuan, WU Qiu-di, ZHANG Li. Developer Recommendation Method for Crowdsourcing Tasks in Open Source Community [J]. Computer Science, 2022, 49(12): 99-108.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220400289

开源社区众包任务的开发者推荐方法

蒋 竞 平 源 吴秋迪 张 莉
北京航空航天大学计算机学院 北京 100191
(jiangjing@buaa.edu.cn)

摘 要 Gitcoin是一个基于开源社区GitHub的众包平台。在Gitcoin中,项目团队可以发布开发任务,开发者选择感兴趣的任务并注册,发布者选择合适的开发者完成任务并发放赏金。但是一些任务因缺乏注册者而失败,部分任务未能合格完成,顺利完成的任务也面临开发者注册间隔时间长的问题。因此,需要一种开发者推荐方法,快速为众包任务发现合适的开发人员,缩短开发者注册众包任务的时间,发现潜在合适的开发者并激励其注册,促进众包任务顺利完成。文中提出了一种基于LGBM分类算法的开发者推荐方法DEVRec(Developer Recommendation)。该方法提取任务特征、开发者特征、开发者和任务的关系特征,使用LGBM分类算法进行二分类,计算开发者注册任务的概率,最终得到众包任务的推荐人员列表。为了评估推荐效果,获取Gitcoin的1599个已完成众包任务、343名任务发布者和1605名开发者。实验结果显示,与对比方法Policy Model相比,DEVRec前1位、前3位、前5位和前10位推荐的准确度及MRR指标分别提高了73.11%,119.07%,86.55%,29.24%和62.27%。

关键词:开源软件;开发者推荐;众包开发;特征提取;机器学习
中图法分类号 TP311

Developer Recommendation Method for Crowdsourcing Tasks in Open Source Community

JIANG Jing,PING Yuan,WU Qiu-di and ZHANG Li
School of Computer Science and Engineering,Beihang University,Beijing 100191,China

Abstract Gitcoin is a crowdsourcing platform based on open-source community GitHub. In Gitcoin,project teams can release development tasks. The developers select the task they are interested in to register,and the publisher selects the appropriate developer to complete the task and offers a reward. But some tasks fail because of a lack of registrants. Some tasks are not performed properly. Successfully completed tasks also face the problem of long developer registration intervals. Therefore,a developer recommendation method is needed to quickly find suitable developers for crowdsourcing tasks,shorten the time for developers to register for crowdsourcing tasks,find potential suitable developers and motivate them to register,so as to promote the successful completion of crowdsourcing tasks. A developer recommendation system DEVRec based on the LGBM classification algorithm is proposed in this paper. Firstly,the task-related characteristics,developer-related characteristics,and the relationship between developers and tasks in the crowd-sourcing task assignment records are extracted. Then the LGBM classification algorithm is used for binary classification. The probability of a developer registering the task is given,and finally the list of recommended people for the task is provided. To evaluate the recommendation effect,1599 completed crowdsourcing tasks,343 publishers,and 1605 developers are crawled from Gitcoin platform. Experimental results show that,compared with the Policy Model,the recommendation accuracy and MRR index of the top 1,top3,top5 and top10 of DEVRec improves by 73.11%,119.07%,86.55%,29.24% and 62.27% respectively.

Keywords Open-source software,Developer recommendation,Crowdsourcing development, Feature extraction,Machine learning

到稿日期:2022-04-28 返修日期:2022-06-10
基金项目:科技创新2030——"新一代人工智能"重大项目(2018AAA0102304);国家自然科学基金(62177003);中央高校基本科研业务费专项资金(YWF-20-BJ-J-1018)
This work was supported by the National Key Research and Development Program of China(2018AAA0102304), National Natural Science Foundation of China(62177003) and Fundamental Research Funds for the Central Universities of Ministry of Education of China(YWF-20-BJ-J-1018).
通信作者:张莉(lily@buaa.edu.cn)

1 引言

GitHub[1-2]是著名的开源软件社区，支持开发者高效参加开源软件开发[3-4]。Gitcoin[5-6]是一个基于开源社区GitHub的众包平台。在Gitcoin中，项目团队可以发布开发任务，开发者选择感兴趣的任务并注册，发布者从注册列表中选择合适的开发者来完成任务并发放赏金。在Gitcoin平台，项目团队采用众包的方式[7-10]，利用赏金激励开发者完成特定任务，推进开源软件项目的开发。

在众包开发实践中，开发者自由参与感兴趣的任务，并把代码发送给发布者审查。一些任务可能由于缺乏开发者关注，未能顺利完成。文献[11]发现，在众包开发平台Topcoder上，87.4%的任务未收到任何开发者提交的内容；在12.6%的收到提交内容的任务中，35.4%的提交内容未能通过发布者的审查。在本文收集的数据集中，12.31%的众包任务未被任何开发者注册；而在顺利完成的众包任务中，任务发布到开发者注册的平均间隔时间达到12.19天。一些众包任务存在缺乏开发者注册、提交质量低、等待注册时间长等问题。这些问题致使众包任务失败，从而影响开源项目的开发进展。因此，众包平台需要一种开发者推荐方法，快速为众包任务发现合适的开发人员。众包平台可以自动向开发人员推荐任务，同时也可以请众包任务的发布者主动联系潜在的注册者，缩短众包任务等待开发者注册的时间，促进众包任务的顺利完成。

现有的一些文献对众包任务的开发者推荐进行了研究[11-18]。文献[16]从开发人员的注册历史中提取特征，并使用基于内容的推荐技术来自动给任务推荐合适的开发人员。文献[19]采用了类似于协同推荐的方法，基于众包任务的历史数据，找出任务之间的共性，然后推荐与新任务相似的历史任务中的开发人员。Zhang等[18]进一步提出了基于任务特征和开发者特征的推荐模型Policy Model，采取元学习的方式推荐开发者。实验结果表明，Policy Model的效果明显优于只考虑任务特征和只考虑开发者特征的方法。然而，现有研究分别提取任务特征与开发者特征，未能关注到开发者与开发任务之间的关系。开发者更可能注册与自己技能相匹配的任务，也可能会和同一任务发布者进行多次合作。因此，开发者和任务之间的关系也可能影响开发者的推荐。

本文提出了一种基于LGBM分类算法的开发者推荐方法DEVRec，提高了开源软件众包任务的注册效率。在Zhang等的研究[18]的基础上，本文不仅扩展了任务特征与开发者特征，而且提出了关系特征度量开发者和任务之间的关系。然后使用LGBM分类算法进行二分类，计算开发者注册任务的概率，最终得到任务的推荐人员列表。为了评估推荐效果，本文获取Gitcoin的1599个已完成众包任务、260名任务发布者和194名开发者。实验结果显示，与Policy Model[18]相比，DEVRec在前1位、前3位、前5位和前10位推荐的准确度及MRR指标分别提高了73.11%，119.07%，86.55%，29.24%和62.27%。

本文的主要贡献包括以下两方面：

(1)分析无人注册任务的特点。与成功任务相比，无人注册任务的奖励较少，发布信息的未知比例更高。

(2)设计众包任务开发人员推荐方法DEVRec。本文分析开源社区众包平台的特点，对现有研究的任务特征、开发者特征进行拓展，添加了任务预计耗时、开发者积压任务数等特征。此外，本文考虑了开发者和任务之间的关系对开发者推荐的影响，提取了开发者和任务之间的关系特征。DEVRec将开发者推荐转化为二分类问题，使用LGBM分类算法为众包任务推荐出合适的开发人员，缩短了开发者注册众包任务的时间，帮助众包任务顺利完成。实验结果说明，DEVRec方法的效果优于对比方法Policy Model。

本文第2节概述了所研究课题已有的一些方法；第3节介绍了本文的研究背景，以及Gitcoin平台的相关知识；第4节详细介绍了本文提出的DEVRec推荐方法的过程和特征选取方式；第5节通过定量实验验证了本文方法的效果；第6节进行了有效性分析；最后总结全文并展望未来。

2 相关工作

近年来，一些研究人员针对众包软件开发平台中的开发者自动推荐，从不同角度进行了研究[16-25]。

文献[16]发现，对于每个任务而言，潜在的开发人员数目非常庞大，如果任务分配不当可能会降低软件交付的质量。其采用CrowdRex推荐方法[16]提取开发者的历史数据作为推荐开发者的依据，将其输入到推荐模型中，将推荐问题转化为多分类问题，使用决策树算法进行多分类求解[16]。

文献[17]提出了CBC-CN方法，该方法首先提取任务的主要特征，如标题、要求等。然后，根据所提取的特征对任务进行聚类，构建分类模型。对于待推荐任务，该方法首先使用分类模型为其推荐出初始的开发者列表，之后构建开发者之间的历史竞争关系网络，并据此对推荐列表中的顺序进行调整。

文献[19]利用开发者的提交率、提交质量、任务的竞争情况等影响因素，使用随机森林分类器，构建了一个动态的人群工作者决策支持模型(DCW-DS)。

Zhang等[18]在前人研究的基础上提出了一种基于任务特征和开发者特征的推荐模型Policy Model。该模型仅将较可能选择任务并提交的开发者加入考虑范围。他们在已有特征的基础上，采取Meta Learning方法来进行开发者推荐。提取的任务的特征包括任务的文本描述、所需的编程语言技术、任务发布日期、任务持续的天数、任务的奖励和任务的难度，开发者的特征包括开发者自身能力以及开发者之间的相互影响。Policy Model的方法推荐准确度优于文献[16-17,19]中的推荐方法。因此，本文选取Policy Model[18]作为主要对比方法。

文献[20]提出了基于活跃度的评论开发人员推荐方法ABRec。针对合适代码发表评论和针对合适任务注册都是基于开发人员的兴趣，也都涉及开发人员和特定内容的知识技能匹配程度。因此，选择ABRec作为本文的对比方法。

3　研究背景

3.1　Gitcoin 众包开发流程

Gitcoin 的开发流程[26-27]如图 1 所示，具体如下：1）任务发布者将任务发布到 Gitcoin 上；2）开发者首先进行任务注册报名；3）任务发布者根据开发者的历史信息和能力，选取匹配的开发者来完成任务；4）开发者需要在规定时间内提交自己的解决方案，如果不提交则视为放弃任务，如果提交了解决方案，任务发布者审核该方案是否满足需求，如果接受方案，则将任务金额发放给开发者。

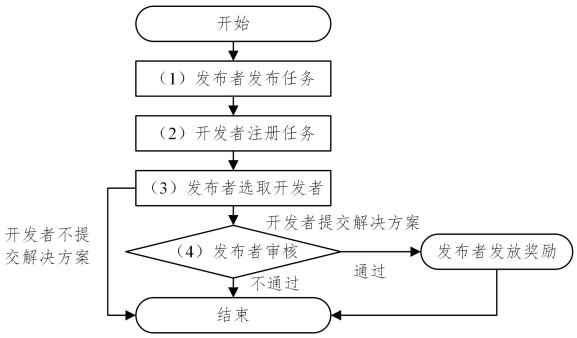

图 1　Gitcoin 众包开发流程

Fig. 1　Crowdsourcing development process in Gitcoin

3.2　Gitcoin 众包任务介绍

本小节在描述 Gitcoin 开发流程的基础上，对 Gitcoin 中的任务进行举例说明。Gitcoin 中一个众包任务的信息主要包括如下部分：

（1）任务标题。任务标题是对众包任务要解决问题的简单概括。

（2）任务奖励。开发者完成任务后获得的奖励，包括现金奖励和虚拟货币奖励。

（3）任务所需技能。该任务涉及的编程语言和技术框架。

（4）任务的相关信息。发布者在发布任务时需要填写的信息包括：任务的类型、任务预计难度、任务预计耗时、任务当前状态等。任务的类型包括：Bug，Feature，Security，Improvement，Documentation，CodeReview，Andere，Design，Other 和 Unknown。任务预计难度包括：Beginner，Intermediate，Advanced，Mittlere 和 Unknown。任务预计耗时包括：Hours，Days，Weeks，Months 和 Unknown。

（5）任务对应的 GitHub 链接。众包任务与 GitHub 中的项目开发相联系，开发者在这里给出了对应项目的链接。

（6）任务文本描述。文本描述包含了任务需要解决的具体问题、任务的需求文档、任务的注意事项等，这是开发者了解任务内容的主要信息来源。

（7）开发者提交解决方案。开发者被选中后提交的解决方案。并非全部开发任务都能完成开发流程，本文获取的数据显示有 12.31%的任务因没有开发者注册而失败。

如图 2 所示，图中的众包任务示例标题为“Use UNO Platform Instead of Ooui”，现金奖励为 72 美元，虚拟货币为 72DAI。任务预计所需等级为“Intermediate”，预计耗时为“Hours”，涉及的编程语言和技术框架包括“C＃，uno，xamarin，xamarinforms，webassembly”。众包任务中显示，开发者 3 个月前提交了解决方案，工作时长为 6 h，提交后的状态为 BountyPaid，说明发布者接受了该方案，完成了全部开发流程。

图 2　Gitcoin 众包任务示例

Fig. 2　Example of crowdsourcing tasks in Gitcoin

3.3　无人注册任务特征分析

在本文获取的 Gitcoin 数据集中，388 个任务因无人注册而失败，占全部任务的 12.31%。为了比较无人注册任务和成功任务，我们统计了多项任务信息，包括虚拟货币奖励、现金奖励、文本描述长度、预计耗时、预计难度和类型等。表 1—表 4 列出了统计结果。

表 1　无人注册任务与成功任务的平均虚拟货币奖励、平均现金奖励、平均文本描述长度情况

Table1　Average virtual currency reward，cash reward，and text description length for unregistered tasks and successful tasks

任务信息	平均虚拟货币奖励/ETH	平均现金奖励/美元	平均描述长度/字符数
无人注册的任务	0.180	65.391	655.111
成功的任务	1.393	336.525	1 232.357

表 2　无人注册任务与成功任务的预计耗时占比

Table 2　Ratio of estimated time spent on unregistered tasks and successful tasks

预计耗时	Hours	Days	Weeks	Months	Unknown
无人注册的任务	0.322	0.173	0.054	0.021	0.430
成功的任务	0.432	0.267	0.056	0.006	0.239

表 3　无人注册任务与成功任务的预计任务难度占比

Table 3　Ratio of task difficulty of unregistered tasks and successful tasks

预计难度	Beginner	Intermediate	Advanced	Mittlere	Unknown
无人注册的任务	0.232	0.247	0.098	0.000	0.423
成功的任务	0.189	0.503	0.078	0.004	0.226

表 4　无人注册任务与成功任务的类型占比

Table 4　Ratio of types of unregistered and successful tasks

任务类型	Documentation	Security	Improvement	Bug	Feature	Andere	CodeReview	Design	Other	Unknown
无人注册的任务	0.015	0.026	0.049	0.095	0.338	0.000	0.000	0.000	0.059	0.418
成功的任务	0.05	0.004	0.189	0.045	0.380	0.001	0.003	0.007	0.066	0.255

由表 1 可以看出，相比成功的任务，无人注册任务的虚拟货币奖励和现金奖励都明显较少，文本描述长度也较短。较少的奖励可能导致任务无人注册。另外，由表 2－表 4 可以看出，对于任务预计耗时、任务难度和任务类型，无人注册任务的未知(Unknown)比例明显高于成功任务。发布者可能缺乏对任务的预估，不确定如何填写相关信息，导致开发者缺乏对任务的了解而不愿意注册。由于任务的发布者与开发者之间沟通对接的不足，适合完成任务的开发者可能没有获知任务信息，导致合适的开发者未能注册任务。本文向任务推荐合适的开发者，辅助任务发布者与开发者之间的对接，促进任务顺利完成。

4 DEVRec 方法设计

4.1 推荐方法总体思路

将众包开发平台中的每一个任务和每一个开发者进行组合，组合中的开发者和任务只有注册和未注册两种关系，因此为任务推荐开发者问题可被视为机器学习的二分类问题。

本文提出了基于机器学习的方法 DEVRec。该方法将开发者推荐问题转化为机器学习的分类问题，考虑了任务特征、开发者特征和开发者与任务的关系特征，并使用 LightGBM 作为分类算法来推荐开发者。

如图 3 所示，整个方法的推荐过程包含两个阶段：训练阶段和推荐阶段。在训练阶段，DEVRec 根据训练数据获取任务特征、开发者特征和开发者与任务的关系特征，输入并训练得到 LGBM 分类模型。在推荐阶段，DEVRec 获取待推荐任务的相关特征，并利用训练好的 LGBM 模型来给任务推荐开发者。

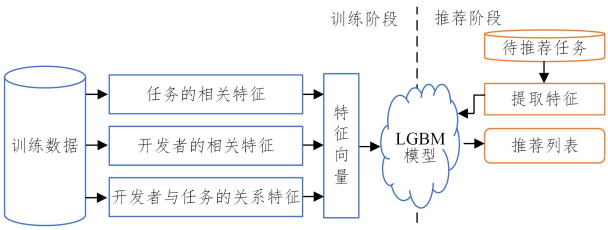

图 3 DEVRec 推荐过程

Fig. 3 Recommendation process of DEVRec

在训练阶段，DEVRec 从训练数据中提取特征向量。本文使用向量空间模型(Vector Space Model)来构建权重向量，根据众包任务的最终注册人员来构建模型的输出标签。对于某一个特征向量，如果对应的开发者实际注册了对应的众包任务，则输出的分类标签为 1，否则为 0。

在推荐阶段，DEVRec 使用训练好的模型为待推荐的任务推荐开发者。首先提取待推荐任务的特征向量，其次计算开发者的特征向量，最后将任务和开发者一一对应并计算关系特征向量。本文方法将以上特征作为分类模型的输入，模型输出每个开发者注册每个任务的概率。对于任务 T，可以得出所有开发者注册任务 T 的概率，并按照开发者注册概率由高到低进行排序，即得到任务 T 的推荐人员列表。

4.2 影响因素选择与量化

表 5 列出了本文提出的指标体系。本文参考 Zhang 等的研究[18]，扩展任务特征与开发者特征。此外，本文提出关系特征来度量开发者和任务之间的关系。

表 5 指标描述

Table 5 Indicator description

	指标名称	是否参考对比方法[18]
任务指标	任务的标题	是
	任务的文本描述	是
	任务所需的编程技能	是
	任务持续的天数	是
	任务涉及的虚拟货币金额	是
	任务涉及的现金金额	是
	任务的难度	是
	任务预计耗时	否
开发者指标	开发者最后一次注册时间	是
	开发者注册任务总数	是
	开发者近 3 个月的注册任务数	否
	开发者积压任务数	否
	近十次任务的注册数	否
	开发者的影响力排名	是
关系指标	开发者技能与任务所需技能的匹配程度	否
	基于时间衰减的开发者与任务发布者的关系	否
	基于时间衰减的开发者历史参与任务与当前任务文本相关性	否
	开发者历史参与任务和当前任务所需技能的相关性	否
	开发者历史参与任务和当前任务的类型相关性	否

4.2.1 任务的相关特征

开发者可以通过众包平台查看众包任务的相关信息，选择自己感兴趣的任务进行注册。任务本身提供了许多可利用的信息，同时任务的一些属性会影响开发人员的注册行为，例如任务的金额和难度等因素。本文主要考虑了以下任务的相关特征：

(1)任务的标题。标题往往会反映任务的主要内容和需求。本文采用了 Word2Vec 技术，对任务的标题进行编码，实现将标题的文本转换为向量。Word2Vec 是一组用于词嵌入的相关模型，它考虑了语义和单词顺序。Word2Vec 使用神经网络技术，将一个大型文本语料库作为输入，每个词语都可以映射成一个特征向量，其中共享公共上下文的词在向量空间中距离更近。为了计算包含 N 个单词的文本向量，本文首先采用 Word2Vec 生成 N 个单词的词向量，再将 N 个单词的词向量相加求平均值，作为文本向量。

(2)任务的文本描述。任务页面的文本描述包含了任务需要解决的具体问题、任务的需求文档、任务的注意事项等，这是开发者了解任务内容的主要信息来源。本文采用了 Word2Vec 技术，将任务的文本描述转换成了向量。

(3)任务所需的编程技能。发布者在发布任务时会写明任务涉及的技术、编程语言和编程框架等。可能包含如 Python 编程语言、Web 网页开发技术、Django 编程框架等关键字。本文涉及到的编程技能共有 687 种。本文采用 One-Hot 编码，对任务所需的技术和编程语言进行了编码。

(4)任务持续的天数。众包任务开发过程包含任务注册、任务提交、任务评审、奖励发放等阶段，这些阶段共同组成了任务的生命周期。此特征为发布者在发布任务时预先设置的整个任务生命周期持续的天数。

(5)任务涉及的虚拟货币金额。即任务给予最后完成者的虚拟货币奖励金额。虚拟货币指众包社区中的虚拟货币。

(6)任务涉及的现金奖励。其指任务给予最后完成者的现金奖励金额。

(7)任务的难度。Gitcoin 上将任务的难易程度划分为 Beginner，Intermediate，Advanced，Mittlere 几个档次，任务发布者在发布任务时需要标明任务的预估难度。本文分别用 1，2，3，4 来代表这 4 种不同难度。

(8)任务预计耗时。任务发布时，发布者会对工作量进行估计，预计开发者要花多长时间来完成任务。Gitcoin 上的预计耗时分为 Hours，Days，Weeks，Months 这 4 种。本文分别用 1，2，3，4 来代表这 4 种不同的任务耗时。

4.2.2　开发者的相关特征

作为众包任务的参与者，不同开发者使用的技术和编程语言各不相同，开发者的历史行为在一定程度上能反映出开发者的开发能力和开发习惯。此外，开发者在不同时间段中表现出的活跃程度也会有所不同，影响其注册众包任务的可能性。因此，开发者自身的属性也是有价值的信息。本文主要考虑了以下开发者的相关特征：

(1)开发者最后一次注册时间。开发者最近一次注册任务的时间距离当前任务发布时间的天数。

(2)开发者注册任务总数。开发者在当前任务发布时间前注册的任务总数。该特征反映了开发者在 Gitcoin 平台上的总体活跃程度。

(3)开发者近 3 个月的注册任务数。在当前任务发布时间的前 3 个月中开发者注册的任务个数。该特征反映了开发者近期的活跃程度。

(4)开发者积压任务数。如果开发者在当前任务发布时间前参与了其他任务的开发和提交，并且其他任务的结束时间大于当前任务的发布时间，那么可以认为开发者手中有积压的未完成任务。当开发者有积压任务未完成时，他可能不会注册新的任务。本文统计在任务发布时间前开发者积压了多少任务还没有完成。

(5)近十次任务的注册数，在当前任务发布的前十个任务中开发者注册的数目。

(6)开发者的影响力排名。开发者的影响力排名是根据 PageRank 算法[18]计算的：首先构建人员关系影响图，图中的边 $IR(A,B)$ 代表开发者 A 对开发者 B 的影响力大小，等于 A 和 B 都注册的任务数量除以 B 注册的任务数量。该边的值越大，B 和 A 选择注册任务的 A 相似度越高，那么未来 B 更可能选择 A 注册的任务，A 对 B 的影响更大。根据 PageRank 算法，人员关系影响图中每个节点的影响力值等于当前节点对其他节点的影响之和，根据节点的影响力值进行排序，得到每个节点的排名，影响力值越大，排名越靠前。每次任务完成后更新人员关系影响图，重新计算开发人员的影响力排名。

4.2.3　开发者与任务的关系特征

除了开发者和任务本身的特征外，开发者和任务之间还存在一些联系。开发者更可能注册与自己技能相匹配的任务，更可能注册一类相似的任务，也可能会与同一任务发布者进行多次合作。因此，开发者和任务之间的关系也会影响开发者的推荐。本文主要考虑了以下特征：

(1)开发者技能与任务所需技能的匹配程度。开发者在注册账号时，可以写明自己擅长的领域和能力。如果开发者填写了相关的信息，那么就可以计算开发者自身编程技能与任务所需的编程语言及技能的匹配程度。例如，任务所需技能为 Web 和 Java，开发者自身编程技能为 C++，Java，Python，那么开发者和任务的匹配度就是 1/2。如果开发者没有填写自己擅长的技能，则记匹配度为 0。开发者 U 的自身的编程技能集合为 $Skill_U$，任务 C 所需的编程技能集合为 $Skill_C$，那么开发者技能与任务所需技能的匹配程度为：

$$\frac{|Skill_U \cap Skill_C|}{|Skill_C|} \tag{1}$$

(2)开发者与任务的历史匹配程度。即开发者历史完成的任务所需技能的集合与当前任务所需技能的相似度。有的开发者没有写明自己擅长的技能，或者填写的信息没有及时更新，这时就需要从他的历史行为中推测开发者和当前任务的匹配程度。定义任务 C 和开发者 U，任务 C 所需的技能集合为 $LSet_C$，开发者 U 的历史注册任务的集合为 $USet_C$，将 $USet_C$ 中每个任务所需的技能集合合并且去掉重复元素，得到开发者 U 的历史注册任务所需技能的集合 $LSet_U$。那么开发者与任务的历史匹配程度为：

$$\frac{|LSet_C \cap LSet_U|}{|LSet_C|} \tag{2}$$

其中，$|LSet_C \cap LSet_U|$ 代表 $LSet_C$ 和 $LSet_U$ 中相同的技能数目，$|LSet_C|$ 代表 $LSet_C$ 中的技能数目。比如 $LSet_C$ 集合为{Java，C，Web，Doc}，$LSet_U$ 集合为{Java，Web}，那么匹配程度为 $2/4=1/2$。

(3)基于时间衰减的开发者与任务发布者的关系。如果开发者历史上曾经注册过当前任务发布者发布的任务，那么开发者更有可能注册当前任务。定义任务的发布者为 R，开发者为 U，任务 C 的创建时间为 t_C，发布者 R 在时间 t_C 之前 α 天内发布的任务集合为 $RSet_C$，开发者 U 在时间 t_C 之前 α 天内注册的任务集合为 $USet_C$，则 $RSet_C \cap USet_C$ 为发布者 R 与开发者 U 在 t_C 之前的任务交集，反映了开发者 U 和发布者 R 的历史关系。那么基于时间衰减的开发者 U 与任务发布者 R 的历史关系为：

$$\sum_{C_j \in (RSet_C \cap USet_C)} (t_C - t_{C_j})^{-1},\ t_C - t_{C_j} < \alpha \tag{3}$$

其中，α 为时间衰减间隔，对于开发者与任务发布者的关系需要按时间间隔 α 进行分段处理。对于发布者发布的一个众包任务，如果其发布时间与待推荐任务的发布时间间隔在 α 天以内，则将其加入到任务集合 $RSet_C$ 中；对于距离待推荐任务的发布时间 α 天以外的众包任务，则不予考虑。

(4)基于时间衰减的开发者历史参与任务与当前任务的文本相关性。即开发者历史完成的任务的文本信息与当前任务文本信息的相似度。文本信息包括任务标题和任务的文本描述。定义开发者 U 和当前任务 C，任务 C 的创建时间为 t_C，开发者 U 在时间 t_C 之前 α 天内注册的任务集合为 $USet_C$。对于任务 $C_j \in USet_C$，获取 C_j 的文本信息、创建时间 t_{C_j}，并通过 Word2Vec 计算 C_j 与 C 的文本向量，利用向量的余弦距离

计算文本相似度S_{Cj}，最后求按时间衰减的平均值，结果如下：

$$\sum_{C_j \in USet_C} S_{Cj}(t_C - t_{Cj})^{-1}, t_C - t_{Cj} < \alpha \tag{4}$$

其中，α为时间衰减间隔，对于开发者与任务的文本历史相关性需要按时间间隔α进行分段处理。对于开发者注册的一个众包任务，如果其发布时间与待推荐任务的发布时间间隔在α天以内，则会将其加入到任务集合$USet_C$中；对于距离待推荐任务的发布时间α天以外的众包任务，则不予考虑。

(5)开发者历史参与任务和当前任务的类型相关性。任务的类型分为 Bug，Feature，Security，Improvement，Documentation，CodeReview，Andere，Design，Other 和 Unknown。开发者历史参与任务和当前任务的类型相关性定义为：与当前任务类型一致的任务在开发者历史参与任务中所占的比例。这一数值体现了开发者历史完成的任务的类型与当前任务类型的相似度。例如一个任务的类型为 Bug，一个开发者历史注册的任务类型分别为 Bug，Bug，Feature，Security，那么该开发者历史参与任务与当前任务类型的相似度为 2/4=0.5。一些能力全面的开发者参与各种类型的任务，对特定类型的专精程度较低。在推荐开发者时，专注特定类型的开发者优于能力全面的开发者。

5 开发者推荐方法性能评估

5.1 数据集

本文通过 Gitcoin 的开放 API，使用 Python 爬虫收集已经结束的 3151 个众包任务。有 973 个失败任务，其中 388 个任务(12.31%)因没有开发者注册而失败。其他的 2178 个众包任务顺利完成，涉及 343 名任务发布者和 1605 名开发者。本文爬取任务的基本信息，包括任务类型、描述文本、任务金额、创建时间、关闭时间、注册者、获胜者等。

其中，本文只研究那些至少注册过 5 次任务的开发者，因为参与次数过少的开发者历史数据很少，计算出的特征难以准确反映开发者的真实水平。过滤后的数据包含 1599 个众包任务，其中任务发布者 260 名，开发者 194 名。通过统计，从任务发布到开发者注册的平均间隔时间是 12.19 天。

5.2 数据集

为了评估最优的机器学习算法性能，本文参考文献[16]，使用准确度@K 指标和 MRR 指标来评估推荐模型的推荐效果(其中，K 为 1，3，5，10)。

对于每个任务 C，如果任务 C 的 Top k 开发者推荐列表 $TL_{C,k}$ 中存在实际的开发者，那么 $hit(TL_{C,k})$ 为 1，否则为 0。其中，$TL_{C,k}$ 表示任务 C 的开发者推荐列表的前 k 个开发者组成的集合。本文可以求出整个测试集上的 Top k 的准确度@K，表达式如下：

$$ACC@K = \frac{\sum_{c \in CSet}^{|CSet|} hit(TL_{c,k})}{|CSet|} \tag{5}$$

其中，$CSet$ 表示所有任务的集合，c 表示众包任务，$|CSet|$ 是整个数据集中的众包任务数目。

准确度@K 只能反映出实际注册的开发者是否出现在了推荐列表 $TL_{C,k}$ 中，本文进一步分析实际注册的开发者在整个推荐列表中的排名 MRR，这是一个更加细致的评估指标。MRR 的定义如下：

$$MRR = \frac{\sum_{c \in CSet}^{|CSet|} \frac{1}{rank(c)}}{|CSet|} \tag{6}$$

其中，$rank(c)$表示实际注册的开发者在推荐列表中的最高排名。假如实际注册的开发者有两名，他们在推荐列表中的排名分别为 2 和 5，那么 $rank(c)$为他们的最高排名 2。MRR 越高，说明推荐效果越好。

为了比较不同机器学习算法的相对优劣，本文定义了提升度 gain 来评估一种算法相对于另一种算法推荐效果提升的幅度。准确度提升度的定义如下：

$$Gain_{ACC@K} = \frac{ACC@K(1) - ACC@K(2)}{ACC@K(2)} \tag{7}$$

MRR 提升度的定义如下：

$$Gain_{ACC@K} = \frac{MRR(1) - MRR(2)}{MRR(2)} \tag{8}$$

其中，$ACC(1)$表示算法 1 的准确度，$ACC(2)$表示算法 2 的准确度，$MRR(1)$表示算法 1 的 MRR，$MRR(2)$表示算法 2 的 MRR。

如果提升度 $gain > 0$，则说明算法 1 的推荐效果优于算法 2，否则算法 2 的推荐效果优于算法 1。

5.3 实验分析

DEVRec 将数据集中的任务数据按照任务发布时间进行排序，发布时间早的数据排序靠前。DEVRec 将时间排序前 90%的数据划分为训练集，后 10%的数据划分为测试集。DEVRec 用训练集中的数据来训练模型，用训练好的模型来对测试集中的任务进行推荐。根据式(5)和式(6)，计算测试集中所有任务的平均准确度和 MRR，从而评估方法的推荐效果。在实验时，默认分类算法为 LGBM，采用调参后的参数。默认的时间间隔为无限制。

本文提出了以下 5 个研究问题，分析结果分别在后文讨论。

RQ1：DEVRec 方法应该使用哪种分类算法？

本文候选的分类算法包括 ExtraTree，XGBoost，FNN，LGBM，这几种机器学习和神经网络算法都适用于本文的场景。分别用这 4 种算法进行分类，根据分类结果选择效果最好的分类算法。

RQ2：DEVRec 分类算法调参前后的结果对比？

分类算法的参数都有一个初始值。模型训练时，参数如果是默认值，则模型的训练效果可能不能达到预期的效果。通常可以通过调整参数使模型能够更好地拟合数据，具有有效的泛化能力。

RQ3：DEVRec 方法中，不同的特征如何影响模型的性能？

本文提出了影响开发者推荐的重要特征，并且希望了解这些特征对 DEVRec 推荐效果的影响。因此，本文进行了特征选择的研究，研究了 3 个方面的特征对结果的影响。本文出于特征构建复杂度以及时间成本考虑，只选用两个方面的特征进行特征组合。通过研究不同的特征组合来研究不同特征对 DEVRec 推荐效果的影响。

RQ4：不同时间间隔 α 的结果比较？

本实验对 DEVRec 的两个时间衰减相关特征的时间间隔阈值进行了处理，如果一个任务的发布时间与待推荐任务的发布时间间隔在 α 天以内，则会用该任务的数据来计算时间衰减特征。根据推荐结果选取效果最好的时间间隔 α。

RQ5：DEVRec 方法和对比方法结果比较？

本文将 DEVRec 与 Policy Model[18] 和 ABRec[20] 进行对比，验证了 DEVRec 方法的推荐效果。

5.4　RQ1 分类算法的选择

本文拟采用的分类算法包括 ExtraTree，XGBoost，FNN，LGBM，其中 ExtraTree 是基于多个决策树的分类器；XGBoost 是一种新兴的基于梯度提升树的集成学习算法；FNN 是一种常见的神经网络；LGBM 是一种基于树的学习算法的梯度提升框架。

通过 4 种分类学习算法训练出的分类器的准确度和 MRR 结果如表 6 和表 7 所列。

表 6　不同机器学习方法的推荐结果

Table 6　Recommendation results of different machine learning methods

算法	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
XGboost	0.163	0.288	0.300	0.369	0.244
ExtraTree	0.163	0.225	0.275	0.344	0.233
FNN	0.038	0.081	0.100	0.156	0.044
LGBM	0.206	0.425	0.513	0.663	0.357

表 7　LGBM 与其他机器学习方法的 Gain

Table 7　LGBM's Gain compared to other machine learning methods

（单位：%）

算法	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
XGboost	26.38	47.57	71.00	79.67	46.31
ExtraTree	26.38	88.89	86.55	92.73	53.22
FNN	442.11	424.69	413.00	325.00	711.36

表 6、表 7 列出了 ExtraTree，XGBoost 和 LGBM 的结果对比。LGBM 在准确度@1、准确度@3、准确度@5、准确度@10、MRR 指标上分别比 ExtraTree 高 26.38%，88.89%，86.55%，92.73%和 53.22%；比 XGBoost 高 26.38%，47.57%，71.00%，79.67%和 46.31%；比 FNN 高 442.11%，424.69%，413.00%，325.00%和 711.36%。由此可以得出结论，选择 LGBM 作为分类算法的性能要大大优于其他几种分类算法，因此本文选择 LGBM 作为最终的分类算法。

5.5　RQ2 算法调参

在使用 LGBM 算法时，默认的参数 $num\_leaves=32$，也就是树的最大叶子数为 32；$max\_depth=-1$，代表树的最大深度不受限制；$learning\_rate=0.05$，代表学习率为 0.05；$feature\_fraction=1$，代表每次新建一棵树时，使用 100% 的特征；$min\_child\_samples=20$，表示单个叶子节点的最小数据量为 20。为了提高准确度，本文采用 GridSearchCV 网格搜索方法进行调参，最终得到的参数为 $num\_leaves=40$，$max\_depth=4$，$learning\_rate=0.01$，$feature\_fraction=1$，$min\_child\_samples=22$。具体结果如表 8、表 9 所列。可以看出，调参后的方法在准确度@1、准确度@3、准确度@5、准确度@10、MRR 指标上比调参前高 94.34%，57.99%，32.22%，26.29%和 45.71%，这说明可以通过调整参数使模型能够更好地拟合数据，推荐效果更好。

表 8　调参前后的推荐结果

Table 8　Recommendation results before and after adjusting parameters

算法	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
调参前	0.106	0.269	0.388	0.525	0.245
调参后	0.206	0.425	0.513	0.663	0.357

表 9　调参后方法相比调参前方法的 Gain

Table 9　Gain of the method after parameter adjusting compared to the method before parameter adjusting

（单位：%）

准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
94.34	57.99	32.22	26.29	45.71

5.6　RQ3 不同特征对推荐效果的影响

为了研究不同特征对 DEVRec 性能的影响，本小节进行特征的组合研究。研究的特征组如下：

(1)任务特征，包括任务持续时间、任务所需的技能、任务的标题、任务的文本描述、任务的难度、任务预计消耗时间、任务的虚拟货币奖励、任务的现金奖励。

(2)开发者特征，包括开发者的注册任务总数、开发者最近一次注册时间、开发者影响关系图排名、最近三个月开发者注册的任务数、开发者注册了但尚未完成的任务数、最近的 10 次任务开发者注册的任务数。

(3)关系特征，包括开发者与任务所需的技能的匹配度、时间衰减的开发者历史注册任务的文本与当前任务的文本相似度、开发者历史注册任务的类型与现在任务的类型的匹配程度、时间衰减的开发者与任务发布者的历史关系、开发者历史注册任务的技能与现在任务的技能的匹配程度。

考虑到方法的时间复杂度，通过两两组合特征组来研究不同特征对 DEVRec 的推荐效果的影响。结果如表 10 和表 11 所列。

表 10　不同特征组合对推荐效果的影响

Table 10　Influence of different feature combinations on recommendation effect

特征组合	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
开发者特征和关系特征	0.138	0.406	0.494	0.619	0.316
任务特征和关系特征	0.150	0.275	0.363	0.506	0.264
任务特征和开发者特征	0.200	0.363	0.456	0.556	0.322
全特征	0.206	0.425	0.513	0.663	0.357

表 11　全特征与不同特征组合的 Gain

Table 11　Gain of total features compared to different combinations of features

（单位：%）

特征组合	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
开发者特征和关系特征	49.28	4.68	3.85	7.11	12.97
任务特征和关系特征	37.33	54.55	41.32	31.03	35.23
任务特征和开发者特征	3.00	17.08	12.50	19.24	10.87

表 10、表 11 列出了不同特征组合的推荐结果对比。全特征包括开发者特征、任务特征、关系特征。全特征在准确

度@1、准确度@3、准确度@5、准确度@10、MRR 指标上分别比开发者特征和关系特征组合高 49.28%，4.68%，3.85%，7.11%和 12.97%；比任务特征和关系特征组合高37.33%，54.55%，41.32%，31.03%和 35.23%；比任务特征和开发者特征组合高 3.00%，17.08%，12.5%，19.24%和 10.87%。

关系特征表示开发者与当前任务之间的联系。由表 10 及表 11 可以看出，与不包含关系特征的推荐方法相比，加入关系特征的推荐方法的准确度明显提高。本文提出的关系特征在开发者推荐中发挥着明显作用，原因是开发者可能参与和自身擅长技能匹配的新任务。因此，未来开源社区众包平台可以提供按照技能分类的任务发布列表，方便开发者快速查看技能匹配的新任务。另外，发布者发布的任务可能具有相似性和相互联系。开发者如果注册过之前的任务，那么也可能对新任务感兴趣。因此，任务发布者也可以重点关注之前合作过的开发者，促进任务的完成。

5.7 RQ4 时间间隔 α 的选择

本文 4.2 节提出的关系特征中，包含基于时间衰减的开发者与任务发布者的关系和基于时间衰减的开发者历史参与任务与当前任务的文本相关性两个特征。这两个特征有一个重要的参数时间间隔 α，对于特征的计算需要按时间间隔 α 进行分段处理。如果一个任务的发布时间与待推荐任务的发布时间间隔在 α 天以内，则会用该任务的数据来计算特征；对于距离待推荐任务 α 天以外的任务，则在计算特征时不予考虑。这样设计是为了弱化距离当前时间比较久远的任务的影响，加强近期的任务数据的影响。

关于时间间隔 α 的取值，本文分别设置为 15，30，45，60，75，90 以及无限制。如果时间间隔 α 为无限制，那么在计算时间衰减时，对于待推荐任务，数据集中所有发布时间早于待推荐任务发布时间的任务都要参与到特征的计算中。具体结果如表 12、表 13 所列。

表 12 不同 α 取值的推荐

Table 12 Recommendation results of different α values

α 取值	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
15	0.175	0.331	0.431	0.631	0.308
30	0.200	0.356	0.438	0.613	0.329
45	0.163	0.400	0.463	0.656	0.323
60	0.175	0.375	0.450	0.650	0.321
75	0.169	0.381	0.469	0.625	0.324
90	0.175	0.350	0.469	0.619	0.319
无限制	0.206	0.425	0.513	0.663	0.357

表 13 无限制与不同 α 取值的 Gain

Table 13 Gain of unlimited compared to different α values

（单位：%）

α 取值	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
15	17.71	28.40	19.03	5.07	15.91
30	3.00	19.38	17.12	8.16	8.51
45	26.38	6.25	10.80	1.07	10.53
60	17.71	13.33	14.00	2.00	11.21
75	21.89	11.55	9.38	6.08	10.19
90	17.71	21.43	9.38	7.11	11.91

表 12、表 13 列出了不同时间间隔 α 的对比结果。由表 12 可以看出，当时间间隔 α 的取值为无限制时，准确度@1、准确度@3、准确度@5、准确度@10、MRR 指标基本都达到了最优值，因此本文取时间间隔 α 为无限制。

5.8 RQ5 与现有方法的对比评估

Zhang 等提出了一种基于任务特征和开发者特征的推荐方法 Policy Model[18]，并在 Topcoder 平台上进行了验证。为了评估 DEVRec 方法的效果，本文在 Gitcoin 平台复现了 Policy Model[18]。由于 Topcoder 平台和 Gitcoin 平台的差异，本文做以下调整。首先，Zhang 等结合任务持续时间、奖励金额和报名人数等因素计算任务的难度。在 Gitcoin 平台，开发者会填写任务难度，因此本文直接使用开发者填写的任务难度，而不是通过多种属性综合计算。其次，本文复现时难以收集到数据，未考虑开发者在上一次任务的表现情况和开发者注册平台的时间。另外，本文也没有考虑任务提交的时间。文献[20]提出的 ABRec 方法依据活跃度为代码评审推荐评论者，本文复现了 ABRec 方法。

表 14 列出了本文 DEVRec 方法和 Policy Model[18] 及 ABRec[20] 的推荐效果。DEVRec 方法在准确度@1、准确度@3、准确度@5、准确度@10 上的结果分别为 0.206，0.425，0.513，0.663，推荐列表前 10 位的推荐准确度达到了66.3%。MRR 结果为 0.357，意味着实际注册开发者在推荐列表中大概排名第三。

表 14 本文方法与对比方法的推荐结果

Table 14 Recommendation results of the proposed method and comparison methods

方法	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
Policy Model	0.119	0.194	0.275	0.513	0.220
ABRec	0.144	0.263	0.300	0.431	0.239
DEVRec	0.206	0.425	0.513	0.663	0.357

表 15 列出了本文 DEVRec 方法相比对比方法的改进效果。结果显示，本文提出的 DEVRec 方法在准确度@1、准确度@3、准确度@5、准确度@10、MRR 指标上相比 Policy-Model 分别提高了 73.11%，119.07%，86.55%，29.24%和 62.27%，相比 ABRec 分别提高了 43.06%，61.60%，71.00%，53.83%和 49.37%。

表 15 本文方法与对比方法的 Gain

Table 15 Gain of the proposed method compared to compariso method

（单位：%）

方法	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
Policy Model	73.11	119.07	86.55	29.24	62.27
ABRec	43.06	61.60	71.00	53.83	49.37

表 16 列出了本文方法的不同特征组推荐结果与 Policy Model 方法的对比结果。Policy Model 方法在推荐中使用开发者特征和任务特征，本文对这两方面特征进行了拓展。如表 16 所列，基于任务特征和开发者特征推荐的准确度@1、准确度@3、准确度@5、准确度@10、MRR 指标达到了 0.2，0.363，0.456，0.556，0.322，而 Policy Model 推荐结果的指标为 0.119，0.194，0.275，0.513，0.220。因此，本文拓展的特征对推荐结果有明显的提升。此外，本文还考虑了开发者和任务之间的关系对开发者推荐的影响，提取了开发者和任务之间的关系特征。在加入本文提出的关系特征后，使用全部

特征的 DEVRec 方法取得了目前最优的推荐结果，准确度@1、准确度@3、准确度@5、准确度@10、MRR 指标分别达到了0.206，0.425，0.513，0.663，0.357，相比 Policy Model 有较大的提升。

表 16　不同特征组与对比方法的推荐结果

Table 16　Recommendation results of different feature groups and comparison methods

方法	准确度@1	准确度@3	准确度@5	准确度@10	*MRR*
Policy Model	0.119	0.194	0.275	0.513	0.220
全特征	0.206	0.425	0.513	0.663	0.357
开发者特征和关系特征	0.138	0.406	0.494	0.619	0.316
任务特征和关系特征	0.150	0.275	0.363	0.506	0.264
任务特征和开发者特征	0.200	0.363	0.456	0.556	0.322

6　有效性分析

6.1　内部有效性威胁

内部有效性威胁与研究中使用的度量和实验中的设置有关。本文的推荐方法提取了开发者特征、任务特征、开发者与任务之间的特征这 3 方面的特征。可能我们还未考虑到其他特征，在未来的研究中我们将进一步分析众包任务分配过程，尝试更多的特征提取方式。发布者在发布任务时提供任务的文本描述，介绍任务内容。但是，文本描述可能不规范或者不准确，从而影响推荐效果。在 Gitcoin 平台，开发者填写的技能可被修改，即开发者擅长的领域和能力。本文数据集中获取的开发者技能是数据爬取时的开发者技能，可能与任务发布时的技能存在差异，影响推荐效果。在实际应用时，系统可以定期收集数据并更新开发者信息，以提高推荐效果。

本文从 Gitcoin 平台收集了 34151 个众包任务，其中 973 个任务是失败的，这些任务的注册开发者不一定适合任务，因此实验数据集删除了 973 个失败任务。接下来，本文删除了注册任务不足 5 次的开发者及其相关任务。因为参与次数过少的开发者历史数据少，计算出的特征未必能准确反映开发者的真实水平。Gitcoin 是一个基于开源社区 GitHub 的众包平台，而新加入 Gitcoin 的开发者可能已经在 GitHub 开发开源项目。虽然 Gitcoin 缺乏新开发者的历史数据，但是 GitHub 可能积累可用数据。在未来工作中，我们将从 GitHub 收集开发者的数据，尝试向任务推荐新加入众包平台的开发者。

本文在提取文本特征时使用了 Word2Vec 方法。在未来工作中，我们将尝试基于 transformer 的特征提取等方法。在提取任务所需的编程技能时，本文使用了 One-Hot 编码方式，这可能会影响推荐方法的效率。在未来工作中，我们还将尝试嵌入的方式表示编程技能并提高效率。同时，我们将继续尝试更多的深度学习方法，以进一步提高推荐效果。

6.2　外部有效性威胁

外部有效性主要体现在研究方法的普遍性上。本文选取了 Gitcoin 众包开发平台进行研究和实验评估。本文的推荐方法应用到其他平台的效果是未知的。在未来工作中，我们希望在更多的平台分析本文方法的效果。本文复现 Zhang 等的方法[18]时，未能实现收集数据以及开发者注册平台时间等个别指标。

结束语　本文发现，与成功任务相比，无人注册任务的奖励较少，发布信息的未知比例更高。接下来，本文针对为 Gitcoin 众包开发平台任务自动推荐开发者，提出了开发者推荐方法 DEVRec。结合相关研究与 Gitcoin 平台的开发流程，提出了影响开发者推荐的 3 类关键特征，即开发者特征、任务特征、任务与开发者之间的关系特征，并提取数据实现量化。在获取的特征的基础上，使用 LGBM 机器学习方法计算开发者被推荐的概率并最终实现开发者推荐。

在从 Gitcoin 平台中爬取的 1599 个众包任务数据上，我们对 DEVRec 方法进行了测试评估，并将结果与现有的众包平台开发者推荐算法进行了对比。实验结果显示，DEVRec 在前 1 位、前 3 位、前 5 位、前 10 位推荐的准确度及 MRR 指标上分别比 Policy Model 方法[18]提高了 73.11%，119.07%，86.55%，29.24%和 62.27%，取得了较好的推荐结果。我们认为，DEVRec 方法在为众包开发任务进行开发者推荐时效果显著，有助于促进众包开发平台任务的完成，提高项目开发效率。

在未来工作中，我们希望开发在线工具，动态监测 Gitcoin 新发布的任务，推荐开发者并尝试联系 Gitcoin，推广开发者推荐工具，根据实际反馈结果调整方法并提升推荐效果，分析本文方法的实用性。此外，我们希望在注册者推荐的基础上，进一步研究获胜者推荐方法，促进任务高效完成。

参 考 文 献

[1] DABBISH L，STUART C，TSAY J，et al. Social coding in GitHub：transparency and collaboration in an open software repository[C]// Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. 2012：1277-1286.

[2] DUBEY A，ABHINAV K，VIRDI G. A framework to preserve confidentiality in crowdsourced software development[C]// 2017 IEEE/ACM 39th International Conference on Software Engineering Companion(ICSE-C). IEEE，2017：115-117.

[3] BAO L，XIA X，LO D，et al. A large scale study of long-time contributor prediction for github projects[J]. IEEE Transactions on Software Engineering，2019，47(6)：1277-1298.

[4] WANG Q Y，XIA X，LO D，et al. Why Is My Code Change Abandoned? [J]. Information and Software Technology，2019，110(JUN.)：108-120.

[5] SAREMI R L，YANG Y E，RUHE G，et al. Leveraging crowdsourcing for team elasticity：An empirical evaluation at Topcoder[C]// 2017 IEEE/ACM 39th International Conference on Software Engineering：Software Engineering in Practice Track (ICSE-SEIP). IEEE，2017：103-112.

[6] ARCHAK N. Money，glory and cheap talk：analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder. com[C]// Proceedings of the 19th International Conference on World Wide Web. 2010：21-30.

[7] SAXTON G D，OH O，KISHORE R. Rules of Crowdsourcing：Models，Issues，and Systems of Control[J]. Information Systems

Management, 2013, 30(1/2): 2-20.
[8] RUI L L, ZHANG P, HUANG H Q, et al. A trust-based incentive mechanism for crowdsourcing [J]. Journal of Electronics Information Technology, 2016, 38(7): 1808-1815.
[9] SAXTON G D, OH O, KISHORE R. Rules of Crowdsourcing: Models, Issues, and Systems of Control[J]. Information Systems Management, 2013, 30(1/2): 2-20.
[10] HASTEER N, NAZIR N, BANSAL A, et al. Crowdsourcing Software Development: Many Benefits Many Concerns[J]. Procedia Computer Science, 2016, 78: 48-54.
[11] 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 (Software Mining). IEEE, 2017: 32-39.
[12] JIANG J, WU Q, CAO J, et al. Recommending tags for pull requests in GitHub[J]. Information and Software Technology, 2021, 129: 106394.
[13] WANG Z Z, SUN H L, FU Y, et al. Recommending crowdsourced software developersin consideration of skill improvement[C]// 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017: 717-722.
[14] BABA Y, KINOSHITA K, KASHIMA H. Participation recommendation system for crowdsourcing contests[J]. Expert Systems with Applications, 2016, 58: 174-183.
[15] JIANG J, LO D, ZHENG J T, et al. Who should make decision on this pull request? Analyzing time-decaying relationships and file similarities for integrator prediction[J]. Journal of Systems and Software, 2019, 154: 196-210.
[16] YANG Y. Code review decision maker recommendation and result prediction research [D]. Beijing: BeiHang University, 2018.
[17] MAO K, YANG Y, WANG Q, et al. Developer recommendation for crowdsourced software development tasks[C]// 2015 IEEE Symposium on Service-Oriented System Engineering. IEEE, 2015: 347-356.
[18] ZHANG Z Y, SUN H L, ZHANG H Y. Developer recommendation for Topcoder through a meta-learning based policy model [J]. Empirical Software Engineering, 2020, 25(1): 859-889.
[19] YANG Y, KARIM M R, SAREMI R, et al. Who should take this task? dynamic decision support for crowd workers[C]// Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. 2016: 1-10.
[20] JIANG J, YANG Y, HE J H, et al. Who should comment on this pull request? Analyzing attributes for more accurate commenter recommendation in pull-based development-Science Direct[J]. Information & Software Technology, 2017, 84(C): 48-62.
[21] HANNEBAUER C, PATALAS M, STÜNKEL S, et al. Automatically recommending code reviewers based on their expertise: An empiricalcomparison [C]// Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. 2016: 99-110.
[22] BEGEL A, HERBSLEB J D, STOREY M A. The future of collaborative software development[C]// Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion. 2012: 17-18.
[23] CUI C, HU M Q, WEIR J D, et al. A recommendation system for meta-modeling: A meta-learning based approach[J]. Expert Systems with Applications, 2016, 46(Mar.): 33-44.
[24] HAUFF C, GOUSIOS G. Matching GitHub developer profiles to job advertisements[C]// 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, 2015: 362-366.
[25] WAN Y, CHEN L, XU G D, et al. SCSMiner: mining social coding sites for software developer recommendation with relevance propagation[J]. World Wide Web, 2018, 21(6): 1523-1543.
[26] GOUSIOS G, ZAIDMAN A, STOREY M A, et al. Work practices and challenges in pull-based development: the integrator's perspective[C]// 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE, 2015: 358-368.
[27] ALELYANI T, YANG Y. Software crowdsourcing reliability: an empirical study on developers behavior[C]// Proceedings of the 2nd International Workshop on Software Analytics. 2016: 36-42.

JIANG Jing, born in 1985, Ph.D, associate professor. Her main research interests include intelligent software engineering, empirical software engineering, open source software and software repository mining.

ZHANG Li, born in 1968, Ph.D, professor. Her main research interests include software modeling and analysis, requirement engineering, empirical software engineering and software architecture.

(责任编辑:喻藜)