软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(5):1635−1651 [doi: 10.13328/j.cnki.jos.006553]

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

一种结合显式特征和隐式特征的开发者混合推荐算法*

于 旭[1], 何亚东[1], 杜军威[1], 王昭哲[1], 江 峰[1], 巩敦卫[1,2]

[1](青岛科技大学 信息科学技术学院, 山东 青岛 266061)

[2](中国矿业大学 信息与电气工程学院, 江苏 徐州 221116)

通信作者: 杜军威, E-mail: djwqd@163.com

摘 要: 现有开发者推荐算法通过对任务和开发者的显式信息进行挖掘, 抽取任务和开发者的显式特征, 完成针对任务的开发者推荐. 然而, 由于显式信息中的描述信息是主观的, 往往是不精确的, 现有基于显式特征的开发者推荐算法性能不够理想. 众包软件开发平台除包含大量不精确的描述信息外, 还包含客观的、较准确的"任务—开发者"成绩信息, 可以有效地推断任务和开发者的隐式特征. 考虑到隐式特征作为显式特征的补充, 将有效缓解描述信息不精确的难题, 提出一种结合显式特征和隐式特征的开发者混合推荐算法. 首先, 利用任务和开发者的平台可见信息充分提取显式特征, 提出面向显式特征的因子分解机 (FM) 推荐模型建模任务、开发者显式特征和相应评分的映射关系. 然后, 利用"任务—开发者"成绩矩阵提取隐式特征, 提出面向隐式特征的矩阵分解 (MF) 推荐模型. 最后, 融合面向显式特征的 FM 推荐模型和面向隐式特征的 MF 推荐模型, 提出多层感知器融合算法. 进一步, 针对冷启动问题, 首先, 基于历史数据, 构建多层感知器模型建模显式特征到隐式特征的映射关系. 然后, 针对冷启动任务或冷启动开发者, 通过任务或开发者的显式特征求解相应的隐式特征. 最后, 基于已训练好的多层感知器融合算法预测评分. 在 Topcoder 软件众包平台的仿真实验表明本文算法相对于对比算法在 4 种不同测试指标上具有明显的优势.

关键词: 软件众包开发; 开发者推荐; 混合推荐算法; 冷启动难题; 多层感知器融合模型; 因子分解机

中图法分类号: TP311

中文引用格式: 于旭, 何亚东, 杜军威, 王昭哲, 江峰, 巩敦卫. 一种结合显式特征和隐式特征的开发者混合推荐算法. 软件学报, 2022, 33(5): 1635–1651. http://www.jos.org.cn/1000-9825/6553.htm

英文引用格式: Yu X, He YD, Du JW, Wang ZZ, Jiang F, Gong DW. Developer Hybrid Recommendation Algorithm Based on Combination of Explicit Features and Implicit Features. Ruan Jian Xue Bao/Journal of Software, 2022, 33(5): 1635–1651 (in Chinese). http://www.jos.org.cn/1000-9825/6553.htm

Developer Hybrid Recommendation Algorithm Based on Combination of Explicit Features and Implicit Features

YU Xu[1], HE Ya-Dong[1], DU Jun-Wei[1], WANG Zhao-Zhe[1], JIANG Feng[1], GONG Dun-Wei[1,2]

[1](College of Information Science & Technology, Qingdao University of Science & Technology, Qingdao 266061, China)

[2](School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China)

Abstract: Existing developer recommendation algorithms extract explicit features of tasks and developers by mining the explicit information of tasks and developers, so as to recommend developers to specific tasks. However, since the description information in the explicit information is subjective and often imprecise, the performance of existing developer recommendation algorithms based on explicit features is not ideal. The crowdsourcing software development platforms not only have a lot of imprecise description information, but also

* 基金项目: 国家自然科学基金 (6217072142, 61773384, 61973180); 山东省自然科学基金 (ZR2021MF092, ZR2019MF014, ZR2018MF007, ZR2019MF033); 山东省重点研发项目 (2018GGX101052).

本文由"领域软件工程"专题特约编辑汤恩义副教授、江贺教授、陈俊洁副教授、李必信教授以及唐滨副教授推荐.

收稿时间: 2021-08-09; 修改时间: 2021-10-09; 采用时间: 2022-01-10; jos 在线出版时间: 2022-01-28

contain objective and more accurate “task-developer” score information, which can effectively infer implicit features of tasks and developers. Considering that implicit features are supplements to explicit features, which will effectively alleviate the problem of imprecise description information, this study proposes a developer hybrid recommendation algorithm that combines explicit features and implicit features. First, the explicit features are fully extracted from the visible information of tasks and the developers on the platform, and the explicit features-oriented factorization machine (FM) recommendation model is proposed to learn the relationship between explicit features of tasks and developers and the corresponding ratings. Then, implicit features are inferred with the "task-developer" rating matrix, and the implicit features-oriented matrix factorization (MF) recommendation model is proposed. Finally, a multi-layer perceptron fusion algorithm is proposed to fuse the explicit features-oriented FM recommendation model and implicit features-oriented MF recommendation model. Further, for the cold-start problem, first, based on historical data, a multi-layer perceptron model is utilized to learn the mapping relationship between explicit features and implicit features. Then, for the cold-start tasks or the cold-start developers, the implicit features are obtained through their explicit features. Finally, the ratings are predicted based on the trained multi-layer perceptron fusion algorithm. The simulation experiment on the Topcoder software crowdsourcing platform shows that the proposed algorithm outperforms the comparison algorithms significantly in terms of four different evaluation metrics.

Key words: crowdsourcing software development; developer recommendation; hybrid recommendation algorithm; cold-start problem; multi-layer perceptron fusion algorithm; factorization machine (FM)

近些年, 越来越多的软件公司开始采取软件众包开发模式来进行软件生产, 该模式是众包理念在软件工程领域的应用. 众包一词源自 2006 年 6 月的《连线》杂志文章《众包的兴起》, 作者 Jeff Howe 在文章中将其定义为“一个组织使用公开招募方式将其工作外包给未定义的网络化劳动力参与的行为”[1]. 众包软件开发作为众包在软件工程领域的应用实例, 基于 Howe 的定义可看作: 由一个未知的、潜在的大型在线工作组以公开召集形式执行任何外部软件工程任务, 包括需求提取、设计、编码和测试等的行为[2]. 通常认为软件众包这种新兴的软件开发模式可以通过增加并行性来缩短上市时间, 并通过灵活的开发能力降低成本和缺陷率.

目前互联网上已存在多种支持软件任务众包模式的众包平台, 如 TopCoder、GetACoder、AppStori、uTest、BugCrowd、Mob4Hire 和 TestFlight. 这些平台专注于软件工程中不同的任务领域, 如 TopCoder 和 GetACoder 平台支持多种类型的软件开发任务, uTest 和 BugCrowd 分别用于软件测试和安全性分析. 尽管不同众包平台关注的领域不同, 但众包模式通常都涉及 3 种不同类型的参与者: 雇主, 他们有需要完成的软件任务; 工人, 他们参与软件任务; 还有平台, 这提供了一个在线市场. 图 1 简要描述了这 3 种不同类型的参与者在众包软件工程中的作用.

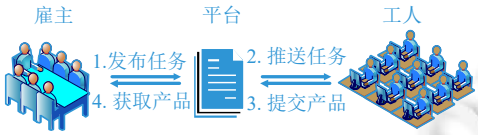


图 1 软件众包模式的不同参与者与作用

随着各个软件众包平台的迅速发展, 发布任务数目与已注册开发者数目急剧增长. 例如, TopCoder 上包含 10 多万个软件项目, 每一天都有成百上千个同时活跃的在线任务, 已注册开发者数目超过 150 万. 众包软件平台上的“信息过载”问题日趋严重, 这使得任务方和开发者群体均面临着严重的选择困难. 从开发者的角度, 尽管他们可以随意选择软件开发任务, 然而面对众包平台上每天发布的众多任务, 选择适合自己的任务是极其困难的. 从任务的角度, 选择有能力的开发者进行开发是保证软件交付质量的关键因素, 然而面对数目庞大的开发者群体, 寻找最可能的开发者也是极具挑战的. 在此背景下, 面向软件众包平台的开发者推荐理论与方法具有重要的理论和应用研究价值, 近些年引起了部分研究者的关注.

Mao 等人[3]首先提取任务特征, 然后利用分类算法将任务特征和获胜开发者的 ID 号进行匹配. Shao 等人[4]首先基于任务特征中的类别和数值属性训练神经网络模型, 然后利用描述属性来训练潜在语义索引 (LSI) 模型, 最后通过结合两个模型来完成开发者的推荐. Zhu 等人[5]针对猪八戒平台的开发者推荐问题, 首先利用潜在狄利克雷分配 (LDA) 主题模型从任务和开发者的平台可见信息 (即显式信息) 中提取少数几个显式特征, 然后采用

排序学习模型进行建模. Zhang 等人[6]提出一种基于元学习的策略模型, 推荐最可能获胜 top k 名开发者. 上述方法仅利用显式信息挖掘开发者和任务的显式特征, 由于显式信息中的描述信息往往是粗糙的、不够精确的, 甚至有夸大的成分, 上述模型的推荐性能是不够理想的.

尽管描述信息会存在一定程度的不准确, 开发者的成绩信息却通常是准确的, 这为准确挖掘任务和开发者的需求和能力相关的隐式特征提供了可能, 其中隐式特征可以看作能力隐空间上的潜在因子向量. 隐式特征是显式特征的有效补充, 而且是客观准确的. 因此, 为解决描述信息含噪声的问题, 本文提出了一种结合显式特征和隐式特征的开发者混合推荐算法 (developer hybrid recommendation algorithm based on the combination of explicit features and implicit features, DHRec). 首先, 利用因子分解机 (FM) 算法[7]建模任务、开发者显式特征和相应评分信息的映射关系, 构建面向显式特征的 FM 推荐模型; 然后, 针对“任务—开发者”成绩矩阵, 基于矩阵分解 (MF) 模型[8]对任务、开发者隐式特征进行建模, 构建面向隐式特征的 MF 推荐模型. 最后, 训练多层感知器模型融合面向显式特征的 FM 推荐模型和面向隐式特征的 MF 推荐模型, 构建多层感知器融合模型. 进一步, 针对冷启动问题, 利用历史数据, 首先建模显式特征和隐式特征之间的映射关系. 然后, 针对冷启动任务或冷启动开发者, 通过其显式特征求解相应的隐式特征. 最后, 基于已训练好的多层感知器融合模型求解. 在 Topcoder 平台数据集进行的广泛对比实验表明, 本文所提算法在 MAE、RMSE、Precision 和 Recall 这 4 个指标上较对比算法具有明显的优势, 显示了本文方法的有效性. 本文主要贡献为:

(1) 给出了针对任务、开发者显式信息的显式特征提取方法, 给出了针对“任务—开发者”成绩矩阵的隐式特征取方法;

(2) 提出了多层感知器模型建模显式特征到隐式特征的映射关系, 从而可以通过冷启动任务或冷启动开发者的显式特征求解相应的隐式特征, 解决了冷启动问题;

(3) 提出了多层感知器融合算法, 实现面向显式特征的 FM 推荐模型和面向隐式特征的 MF 推荐模型的有效融合.

本文第 1 节回顾了开发者推荐相关的研究工作. 第 2 节提出了一种结合显式特征和隐式特征的开发者混合推荐算法. 第 3 节在 Topcoder 软件众包开发数据集上进行了广泛的对比实验, 给出了详细的对比实验结果, 并对实验结果进行了详细分析. 第 4 节对全文进行总结与展望.

1 相关研究

软件众包开发者推荐是传统推荐算法在众包软件工程领域的应用, 因此本节将首先回顾传统推荐算法相关研究, 然后介绍当前软件众包开发者推荐相关研究.

1.1 传统推荐系统

推荐算法以解决“信息过载”问题和为用户提供个性化服务为目标, 近些年得到了国内外学者广泛研究, 主要分为基于内容的推荐[9−11]和协同过滤[12]两种方法. 基于内容的算法根据用户过去喜欢的项目的属性信息为用户推荐相似的项目. Balabanović 等人[9]对基于内容的推荐进行了首次研究. Oord 等人[10]提出了一个基于内容的音乐推荐框架. 由于基于内容的推荐算法通常需要进行项目特征的表示, 不准确或不充分的特征表示将严重影响推荐性能. 相比之下, 基于协同过滤的算法不依赖于项目的内容, 而只使用用户对项目的反馈数据进行用户偏好挖掘, 在推荐系统领域占有主导地位. 过去 20 年, 协同过滤算法获得了全面的研究, 涉及到基于内存的方法[13]和各种各样的基于模型的方法, 如矩阵分解模型[14,15]、概率模型[16,17]和深度学习模型[18,19]等. 然而传统协同过滤算法明显的不足就是难以处理评分矩阵的稀疏性和新用户的冷启动问题, 这严重影响了协同过滤算法的性能.

近年来国内外部分学者尝试利用各种辅助信息, 如社交关系[20,21]、地理位置[22,23]和评论信息[24,25]等, 克服推荐系统的稀疏性和冷启动难题, 其中李晨亮等人[24]提出一种基于胶囊网络的模型 CARP, 该模型可以基于用户评论来预测评分. 另有部分学者进行跨域推荐算法的研究[25−35], 即从含有较为丰富评分数据的辅助域上迁移

有用的知识到评分数据较为匮乏的目标域上, 改善目标域上推荐算法的性能. 此外, 近几年, 综合两类推荐方法的混合推荐算法[36,37]以及基于深度学习技术的推荐算法[38,39]也获得了广泛研究, 为推荐算法领域提供了新的研究思路.

虽然传统推荐算法近些年得到了广泛的应用, 但是传统推荐算法的应用场景往往与本文研究的开发者推荐场景存在显著差异. 一方面, 在传统推荐系统中, 比如电影或者音乐推荐系统, 考虑到隐私问题, 用户通常不情愿透露较多的个人信息. 然而, 在众包软件开发平台上, 由于任务方想获得更高质量的产品, 开发者想得到更多的工作机会, 他们愿意主动提供相对较多的个人信息. 另一方面, 开发者推荐系统相对于传统推荐系统, 冷启动问题的解决显得尤为重要. 因为传统推荐系统的服务对象可以是新用户, 也可以是有过历史行为的老用户, 但是开发者推荐系统的服务对象只能是新任务, 对于已经完成的任务, 平台不会为其再次推荐开发者. 表 1 对传统推荐场景和软件众包平台开发者推荐场景的不同点进行了总结. 因此, 简单应用传统推荐算法不能有效解决软件众包平台开发者推荐问题. 近些年, 面向软件众包平台的开发者推荐, 部分学者也进行了初步研究, 详见第 1.2 节.

表 1 开发者推荐系统与传统推荐系统的不同之处

对比角度	开发者推荐系统	传统推荐系统
个人信息	含个人信息多	含个人信息少
服务对象	仅有新任务	新、老用户均可

1.2 众包软件工程开发者推荐

关于软件众包开发平台的开发者推荐, 最早的系统性的工作来自于 Mao 等人[3], 他们提出一种基于内容的推荐模型, 该模型首先提取任务特征, 然后根据开发者的历史获胜数据, 采用分类模型建立任务特征与开发者 ID 号的映射关系. Shao 等人[4]提出 NNLSI 模型, 首先根据任务特征的类别和数值属性训练神经网络模型, 然后利用任务描述属性训练潜在语义索引 (LSI) 模型, 最后结合两个模型完成开发者推荐. 然而由于冷启动开发者的 ID 号并不存在于训练集的标记集合中, 上述两种算法均不能解决开发者冷启动问题.

Zhu 等人[5]提出一种基于排序学习的开发者推荐模型, 首先使用 LDA 主题模型从任务和开发者的显式信息中提取部分特征, 然后使用排序算法来建立推荐模型. 谢新强等人[40]挖掘开发者与任务的动态交互行为、静态匹配行为、静态匹配度, 以及开发者能力 4 个不同维度的特征, 并结合矩阵分解技术, 提出一种能力与行为感知的多特征融合协同过滤开发者推荐方法. Zhang 等人[6]提出一种基于元学习的策略模型, 推荐最可能获胜 top k 名开发者. 此外, 考虑到开发技能会随着开发经验的积累而提高, Wang 等人[41]研究了开发者技能增长规律, 并提出了一种技能相关的开发者推荐模型. 考虑到开发者之间的竞争关系, Fu 等人[42]提出一种竞争相关的开发者推荐模型. 上述几种方法仅利用了开发者和任务的不够精确的描述信息, 未能充分挖掘客观的较为真实的成绩信息, 推荐性能不够理想.

除了针对任务的开发者推荐外, 目前还有一些针对开发者进行任务推荐的研究[43−46]或跨众包平台推荐研究[47,48]. 由于它们不是本文研究的场景, 此处不再详细介绍.

2 一种结合显式特征和隐式特征的开发者混合推荐算法

本文提出一种结合显式特征和隐式特征的开发者混合推荐算法, 该算法包含多层感知器融合模块 (图 2(a)–图 2(c)) 和多层感知器映射关系建模 (图 2(d)) 两个子模块. 图 2(a) 基于显式信息提取显式特征, 建立 FM 模型; 图 2(b) 基于评分信息提取隐式特征, 建立 MF 模型; 图 2(c) 通过多层感知器模型, 实现两类模型的融合. 图 2(d) 基于历史数据, 构建多层感知器模型建模显式特征到隐式特征的映射关系.

本文结合显式特征和隐式特征的开发者混合推荐算法, 针对冷启动任务或冷启动开发者, 首先基于多层感知器映射关系建模, 通过任务或开发者的显式向量求解相应的隐式特征. 然后, 基于多层感知器融合模块预测评分. 下面将对本文所提算法进行详细介绍.

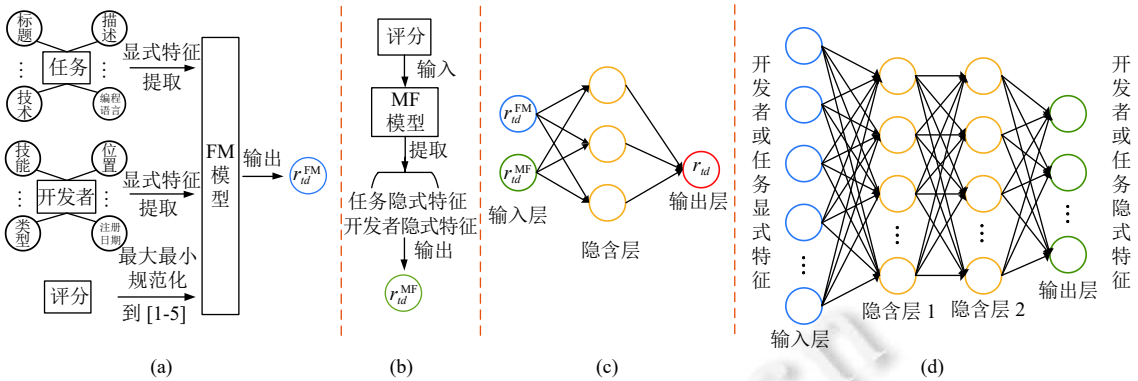


图 2 多层感知器融合模块

2.1 问题描述

众包软件工程场景包含任务集合、开发者集合和任务开发者成绩矩阵 3 类对象, 其中任务集合包括已有任务 $\{t_1, t_2,\ldots, t_{m1}\}$和新任务 (即冷启动任务)$\{t_{m1+1},\ldots, t_{m1+m2}\}$, 开发者集合包括已有开发者$\{d_1, d_2,\ldots, d_{n1}\}$和新开发者 (即冷启动开发者)$\{d_{n1+1},\ldots, d_{n1+n2}\}$, 注意任务和开发者均包含显式信息. 成绩矩阵 $R(m_{1\times}n_1)$ 则包括参与任务的开发者的得分. 图 3 为软件众包平台上的成绩矩阵, 其中我们用分数表示已知成绩, 用问号表示未知成绩. 此外, T_{new} 和 D_{new} 分别表示新发布的任务和新注册的开发者. 众包软件工程开发者推荐问题是指通过分析平台上已有的数据, 从包含冷启动开发者在内的所有开发者集合内, 为冷启动任务分配最为合适的开发者, 下文将对该问题展开深入的分析.

Task \ Developer	d_1	d_2	d_3	d_4	d_5	...	d_n	D_{new}
t_1	?	5	?	2	?	...	1	
t_3	3	?	4	?	2	...	?	
t_3	?	?	4	?	5	...	?	?
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	
t_m	1	3	?	5	?	...	?	
T_{new}	?							?

图 3 众包软件平台举例

2.2 显式特征表示

2.2.1 任务显式特征表示

软件众包任务包含从文本型到数值型等多种类型的特征. 文本型特征主要包括“标题” “任务描述” “编程语言”和“技术”; 数值型特征主要包括“任务发布日期”“持续时间”和“奖励”. 表 2 列出了有关这些特征的详细信息, 接下来, 我们将给出特征的表示和计算方法.

表 2 任务显式特征

任务特征	格式	描述
标题	文本型	发布任务的标题
任务描述	文本型	任务发布者对任务的需求描述
技术	文本型	任务所需要的技术要求
编程语言	文本型	任务所需要的编程语言
任务发布日期	数值型	任务的发布时间
持续时间	数值型	分配给开发者的时间
奖励	数值型	获胜者获得的奖励

(1) 标题和任务描述

由于 Bert 模型[49]近年来在许多自然语言处理领域具有明显的优势, 其输出的编码信息很大程度上保留了文本的语义, 本文利用 Bert 模型对标题和任务描述进行编码. 此外, 我们使用自编码器进行降维处理.

(2) 技术和编程语言

“技术”和“编程语言”特征取值由平台中提供的部分标签组成, 我们使用与标签集中元素数量相等的二进制向量来表示, 其中 1 表示开发者具有该标签, 0 则表示开发者不具备该标签. 例如, 编程语言对应的标签集为{“C ++”, “Java”, “Python”, “Scala”, “Ruby”}, 则标签为{“C ++”, “Python”, “ Ruby”}的开发者可以表示为 (1, 0, 1, 0, 1).

(3) 任务发布日期、持续时间和奖励

以上 3 个数值特征取值可以直接从众包平台获得, 为了消除不同量纲的影响, 采用 Z-score 归一化方法对数值特征值进行处理.

2.2.2 开发者显式特征表示

开发者的显式特征包括基本注册信息, 例如“自我描述”“技能”“开发者类型”“位置”和“注册日期”. 表 3 列出了有关这些特征的详细信息, 接下来, 我们将给出这些特征的表示和计算方法.

表 3 开发者显式特征

开发者特征	格式	描述
自我描述	文本型	开发者的自我描述信息
技能	文本型	开发者擅长的技能
位置	分类型	开发者工作所在的城市
开发者类型	分类型	数据科学、开发、设计
注册日期	数值型	开发者加入平台的时间

(1) 技能

与任务的“技术”和“编程语言”特征一样, 我们也利用二进制向量来表示它.

(2) 自我描述

自我描述信息的表示方式与任务的“标题”和“任务描述”特征相同.

(3) 开发者类型和位置

我们使用 One-hot 编码来表示这两个分类型特征.

(4) 注册日期

注册日期是数值型特征, 与任务的数值型特征预处理方法相同.

2.3 隐式特征表示

由于显式信息中的描述信息往往不够准确、不够充分甚至有夸大的成分, 通过描述信息提取到的部分显式特征也不够准确, 从而无法准确建模显式特征到成绩的关系模型. 为弥补描述信息在准确性上的不足, 考虑到成绩信息是客观的、真实的, 本文将利用成绩信息来进一步挖掘任务和开发者的特征, 我们将这部分特征定义为隐式特征. 隐式特征可以辅助显式特征提升推荐算法的性能.

本文利用 Funk-SVD 矩阵分解模型[8]对成绩矩阵进行分解, 通过将任务和开发者投影到一个共享的潜在因子空间实现任务和开发者的隐式特征提取. 为了使隐式特征更容易被理解, 下面给出一个具体的例子. 假设潜在因子空间为 3 维能力空间, 如“理解产品的能力”“代码实现的能力”“排查错误的能力”, 那么开发者 d 在任务 t 上获得的成绩取值往往与开发者所具有的能力分布 $q_d \in \mathbb{R}^3$ 以及要完成的任务所需要的能力分布 $p_t \in \mathbb{R}^3$ 密切相关. 显然, 当两种分布相匹配时, 开发者能够获得较好的成绩. 通常, 成绩的取值可以看作是两种分布的内积, 即 $r_{td} = q_d^{\mathrm{T}} p_t$.

Funk-SVD 通过低秩分解可以有效缓解数据稀疏性问题, 通过求解如下优化问题 (1) 来获取任务和开发者潜在因子:

$$\min_{q^*,p^*}\sum_{(t,d)\in\kappa}(r_{td}-q_d^{\mathrm{T}}p_t)^2+\lambda(\|q_d\|^2+\|p_t\|^2) \tag{1}$$

其中, κ 是对应已知评分的“任务-开发者”对集合, $\lambda>0$ 为正则化系数, 避免过拟合, $p_t\in\mathbb{R}^k$ 和 $q_d\in\mathbb{R}^k$ 为任务和开发者潜在因子, 本文将其称为任务和开发者的潜在特征. 通常基于交叉验证方法[50]获取参数 λ 和 k 的最优取值.

通常使用随机梯度下降算法求解该优化问题, 迭代公式如下:

$$\begin{aligned}q_d&=q_d+\gamma(e_{td}p_t-\lambda q_d)\\p_t&=p_t+\gamma(e_{td}q_d-\lambda p_t)\end{aligned} \tag{2}$$

其中, $e_{td}=r_{td}-q_d^{\mathrm{T}}p_t$, γ 为学习速率.

2.4 多层感知器的融合模型

2.4.1 面向显式特征的 FM 推荐算法

在显式特征的基础上, 我们利用 FM 算法学习任务—开发者显式特征与评分之间的映射关系. 由于高阶的 FM 模型计算复杂度过高, 本文选择应用更为普遍的二阶 FM 模型进行建模. 二阶 FM 模型也被称为标准 FM 模型, 其表达式见公式 (3).

$$r=w_0+\sum_{i=1}^{N+M}w_ix_i+\sum_{i=1}^{N+M}\sum_{j=i+1}^{N+M}w_{ij}x_ix_j \tag{3}$$

其中, 模型参数 w_0, w_j 和 w_{ij} 分别表示全局偏置, 特征 i 对应的权重和特征 i 与特征 j 交互项的权重, N 表示任务显式特征的维数, M 表示开发者显式特征的维数.

与传统多项式回归不同, FM 算法通过将交互项的权重分解为隐因子向量的乘积来缓解稀疏性问题, 在高维稀疏数据上具有很好的性能. 公式 (3) 中交互项 x_i 和 x_j 的权重 w_{ij} 可以表示为 $w_{ij}=v_i^{\mathrm{T}}v_j$, 其中, v_i 和 v_j 分别表示特征 x_i 和 x_j 对应的隐因子向量. 隐因子向量的维数 k 通常基于 10 折交叉验证方式确定.

为训练 FM 回归模型, 我们通常基于方差定义损失函数, 并求解如下优化问题:

$$(w_0^*,w^*,V^*)=\underset{w_0,w,V}{\operatorname{argmin}}\left(\sum_{(x,y)\in S}(\hat{y}(x|w_0,w,V)-y)^2+\lambda_{w_0}w_0^2+\lambda_w\sum_{i=1}^{N}w_i^2+\lambda_V\sum_{i=1}^{N}\sum_{f=1}^{h}V_{i,f}^2\right) \tag{4}$$

其中, S 表示训练集, λ_{w0}、λ_w、λ_V 分别表示 3 类模型参数对应的正则化系数. 为进一步降低参数调优的复杂度, 将 λ_{w0}、λ_w、λ_V 参数统一为 λ, 此时目标函数为:

$$(w_0^*,w^*,V^*)=\underset{w_0,w,V}{\operatorname{argmin}}\left(\sum_{(x,y)\in S}(\hat{y}(x|w_0,w,V)-y)^2+\lambda\left(w_0^2+\sum_{i=1}^{N}w_i^2+\sum_{i=1}^{N}\sum_{f=1}^{h}V_{i,f}^2\right)\right) \tag{5}$$

对于优化问题 (5), 本文利用随机梯度下降法求解权重参数, 迭代公式如下:

$$\begin{cases}w_0\leftarrow w_0-2\eta((\hat{y}(x|w_0,w,V)-y)+\lambda w_0)\\w_i\leftarrow w_i-2\eta((\hat{y}(x|w_0,w,V)-y)x_i+\lambda w_i)\\V_{i,h}\leftarrow V_{i,h}-2\eta\left((\hat{y}(x|w_0,w,V)-y)x_i\sum_{j\neq i}V_{j,h}x_j+\lambda V_{i,h}\right)\end{cases} \tag{6}$$

其中, η 为学习率.

在求得 (w_0^*,w^*,V^*) 后, 通过公式 (3) 即可预测未知的成绩值. 然而由于部分显式特征往往不够准确, 面向显式特征的 FM 推荐模型性能不够理想. 考虑到成绩信息是客观的、真实的, 成绩信息可以看作是描述信息的有益补充, 因此我们可以进一步通过成绩信息获取的隐式特征, 构建面向隐式特征的矩阵分解推荐模型, 并作为面向显式特征的 FM 推荐模型的有效补充.

2.4.2 面向隐式特征的 MF 推荐算法

在第 2.3 节中, 基于矩阵分解模型, 我们根据成绩矩阵获取了已发布任务和已有开发者的隐式特征 p_t 和 q_d, 在此基础上利用公式 (7) 即可进行成绩预测:

$$\hat{r}_{td}=q_d^{\mathrm{T}}p_t \tag{7}$$

虽然成绩信息是客观的、真实的, 但该信息往往是稀疏的, 所以仅仅依赖成绩信息构建面向隐式特征的矩阵

分解推荐模型, 也不能获得满意的性能. 因此, 本文将给出一种基于多层感知器的融合模型, 以实现上述两种模型的有效融合.

2.4.3 基于多层感知器的融合算法

如上所述, 面向显式特征的 FM 推荐算法 M_1 和面向隐式特征的矩阵分解推荐算法 M_2 均能进行评分预测, 但独立预测的性能不够理想. 为更好地进行成绩预测, 我们提出一种基于多层感知器的融合算法, 如图 2(c) 所示, 输入层包含两个特征变量 x_1, x_2, 分别表示 FM 推荐模型和 MF 推荐模型的独立预测结果, 隐含层的节点数 h 作为参数可以通过交叉验证进行调优. 由于是一个多分类问题, 输出层采用 Softmax 激活函数. 对于给定的样本输入 $\mathbf{x}^{(i)} \in R^h$, Softmax 激活函数能够针对每一类别 j 计算出概率值 $p(y = j|\mathbf{x}^{(i)})$, ($j$=1, 2, 3, 4, 5), 即估计 $\mathbf{x}^{(i)}$ 归属每一类别的概率. 计算公式如下:

$$S_\theta(\mathbf{x}^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|\mathbf{x}^{(i)};\theta) \\ p(y^{(i)} = 2|\mathbf{x}^{(i)};\theta) \\ \vdots \\ p(y^{(i)} = 5|\mathbf{x}^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{5} \mathrm{e}^{\theta_j^T \mathbf{x}^{(i)}}} \begin{bmatrix} \mathrm{e}^{\theta_1^T \mathbf{x}^{(i)}} \\ \mathrm{e}^{\theta_2^T \mathbf{x}^{(i)}} \\ \vdots \\ \mathrm{e}^{\theta_5^T \mathbf{x}^{(i)}} \end{bmatrix} \tag{8}$$

其中, $\theta = (\theta_1, \ldots, \theta_5)^t$, $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_5 \in R^h$ 为对应不同类别的隐含层到输出层的连接权重, $1/\sum_{j=1}^{k} \mathrm{e}^{\theta_j^T x^{(i)}}$ 是归一化概率分布.

2.4.4 多层感知器的融合模型训练过程描述

如上, 我们已经给出了本文多层感知器融合模型的 3 个组成部分, 即面向显式特征的 FM 推荐算法、面向隐式特征的 MF 推荐算法和基于多层感知器的融合算法. 本小节将结合训练数据使用情况详细描述各组成部分的训练过程.

(1) FM 和 MF 训练过程

对于初始训练集, 首先将每行评分随机划分为 k 份, 进行 k 折交叉验证, 确定 FM 和 MF 最优参数. 然后, 利用初始训练集对于最优参数的 FM 和 MF 模型进行训练.

(2) 基于多层感知器的融合算法训练过程

首先, 构造基于多层感知器的融合模型的训练数据如下:

1) 将每行评分随机划分为 k 份, 每次取其中 k–1 份评分数据训练 FM 和 MF, 针对剩下的一份评分数据进行预测. 如此轮换 k 次, 并将 FM 和 MF 预测评分和真实数据合并构造训练集. 显然, 最终的训练集样本个数与初始评分个数一致.

2) 针对上述训练集进行交叉验证操作, 确定基于多层感知器的融合模型的隐含层神经元节点个数的最优值, 并基于最优值在整个训练集上训练基于多层感知器的融合模型.

2.5 冷启动情况分析

尽管我们已经完成多层感知器的融合模型构建, 但是该模型尚无法解决冷启动问题, 这是因为传统 MF 模型在训练过程中无法建模冷启动任务的隐式特征, 从而无法直接利用公式 (7) 求解冷启动问题. 因此, 获取冷启动任务的隐式特征潜在因子是问题解决的关键. 考虑到任务的显式特征与隐式特征是看待任务的两个视角, 往往具有较强的相关性. 基于这一思路, 首先建模显式特征到隐式特征映射关系, 进而基于映射关系根据新发布任务的或新注册开发者的显式特征计算相应的隐式特征, 从而利用多层感知器的融合模型完成评分预测.

2.5.1 基于多层感知器的显式特征到隐式特征映射建模

我们将基于历史数据获取的任务或者开发者的显式特征和隐式特征作为监督训练数据集的输入和输出向量, 训练多层感知器模型, 实现显式特征到隐式特征的映射关系建模, 如图 2(d) 所示.

多层感知器模型损失函数可以定义为:

$$\frac{1}{m} \sum_{i=1}^{m} \left\| \hat{V}_i - V_i \right\|^2 \tag{9}$$

其中, V_i 为任务 i 的隐式特征值, $\hat{V}_i$ 为经过深度回归模型预测的任务 i 的隐式特征值, m 为已有任务的个数.

2.5.2 冷启动任务上已有开发者的成绩预测

在第 2.4.3 节中, 我们提出了基于多层感知器的融合算法, 基于该算法我们可以求解融合后的评分 $r=F(x_1, x_2)$, 其中 x_1 是面向显式特征的 FM 模型所得到的评分, x_2 是面向隐式特征的 MF 推荐算法所得到的评分. 为了使用基于多层感知器的融合算法预测冷启动任务 t_{new} 上开发者 d 的成绩, 首先需要计算冷启动任务 t_{new} 上面向隐式特征的 MF 推荐算法对开发者 d 的成绩的预测值, 根据公式 (7) 可知

$$\hat{r}_{t_{\text{new}}d} = q_d^{\mathrm{T}} p_{t_{\text{new}}} \tag{10}$$

其中, $p_{t_{\text{new}}}$ 可以通过多层感知器映射关系建模进行预测. 然后, 将 $\hat{r}_{t_{\text{new}}d}$ 代入基于多层感知器的融合算法, 即可预测冷启动任务 t_{new} 上开发者 d 的成绩.

2.5.3 冷启动任务上新注册开发者的成绩预测

与冷启动任务上已有开发者的成绩预测类似, 使用该混合模型预测冷启动任务 t_{new} 上开发者 d_{new} 的成绩时, 首先计算成绩的预测值如下:

$$\hat{r}_{t_{\text{new}}d_{\text{new}}} = q_{d_{\text{new}}}^{\mathrm{T}} p_{t_{\text{new}}} \tag{11}$$

其中, $p_{t_{\text{new}}}$ 和 $q_{d_{\text{new}}}$ 均通过多层感知器映射关系建模进行预测. 然后, 将 $\hat{r}_{t_{\text{new}}d_{\text{new}}}$ 代入基于多层感知器的融合算法, 即可进行评分预测.

2.6 算 法

算法 1 给出结合显式特征和隐式特征的开发者混合推荐算法 (DHRec). 为了描述方便, 算法中仅给出冷启动任务对于冷启动开发者的评分预测方法.

算法 1. 结合显式特征和隐式特征的开发者混合推荐算法.

输入: 任务-开发者-评分矩阵 R, 已有任务信息 Info_T$_i$(i= 1,…, m), 冷启动任务信息 Info_New_T$_t$(t=1,…, m_1), 已有开发者信息 Info_D$_j$(j=1, …, n), 冷启动开发者信息 Info_ New_D$_d$(d=1,…, n_1);
输出: Top-k 开发者推荐列表.

1. [$Task_i$, $Task_t$] = Task_feature_extraction (Info_T$_i$ | i=1, …, m. Info_New_T$_t$ | t = 1, …, m_1)
//提取已有任务和冷启动任务的特征 $Task_i$ 和 $Task_t$
2. [Dev_j, Dev_d] = Developer_feature_extraction(Info_D$_j$ | j=1, …, n. Info_ New_D$_d$ | d =1,…, n_1)
//提取已有开发者和冷启动开发者的特征 Dev_j 和 Dev_d
3. [$\hat{w}_0$, $\hat{w}_i$, $\hat{v}_i$] = FM(R, $Task_i$, Dev_j)
//在已知评分矩阵上基于交叉验证训练 FM 模型, 并求得最终的 FM 参数 $\hat{w}_0$, $\hat{w}_i$, $\hat{v}_i$, 其中 i=1, ..., N
4. $\hat{r}_{td}^{FM}$ = FM($Task_t$, Dev_d, $\hat{w}_0$, $\hat{w}_i$, $\hat{v}_i$)
//利用 FM 模型预测冷启动任务对冷启动开发者的评分 $\hat{r}_{td}^{FM}$
5. [U_i, V_j] = MF(R)
//在已知评分矩阵上基于交叉验证训练 MF 模型, 并求得现有任务和现有开发者的隐因子向量, U_i, V_j , 其中 i= 1, …, m. j=1, …, n
6. T=Constructing_training_set(R, $\hat{w}_0$, $\hat{w}_i$, $\hat{v}_i$, U_i, V_j)
//构造训练集 T
7. [W, B]=MLP1(T)
//在训练集 T 上训练基于多层感知器的融合模型
8. [F_1, F_2] = MLP2($Task_i$, Dev_j, U_i, V_j)
//利用多层感知器模型求得任务显式特征和隐式特征映射关系 F_1, 以及开发者显式特征和隐式特征映射关系 F_2
9. U_t =F_1($Task_t$)

//利用映射关系 F_1 求得冷启动任务的隐式特征 U_t, 其中 $t=1,\ldots,m_1$

10. $V_d=F_2(Dev_d)$

//利用映射关系 F_2 求得冷启动开发者的隐式特征 V_d, 其中 $d=1,\ldots,n_1$

11. $\hat{r}_{td}^{MF}=\mathrm{MF}(U_t, V_d)$

//利用 MF 模型预测冷启动任务对冷启动开发者的评分 $\hat{r}_{td}^{MF}$

12. $\hat{r}_{td}=\mathrm{MLP\ 1}(\hat{r}_{td}^{FM}, \hat{r}_{td}^{MF}, \mathrm{W}, \mathrm{B})$

//将 FM 模型和 MF 模型求得的冷启动任务对冷启动开发者的评分代入多层感知器融合模型求得评分

3 实 验

在本节中, 我们在 Topcoder 数据集上进行广泛的对比实验来证明所提 DHRec 算法的有效性. 实验环境配置为 4.7 GHz、i7-10710U CPU、16 GB RAM 和 Windows8. 这些算法是用 Python 3.0 在 3 个开源机器学习库 Gensim、scikitlearn 和 Keras 上实现的. 本实验我们主要讨论如下两个问题:

(1) 本文算法的各个子模型在不同训练集上是否具有不同的最优参数组合?

(2) 本文方法通过结合显式特征与隐式特征是否能够获得更好的推荐性能?

3.1 Topcoder 数据获取介绍

TopCoder 平台的任务大多以竞赛形式发布, 本实验中, 我们爬取软件开发相关的竞赛任务数据, 包括系统设计、代码编写、模块测试等. 第 1 步, 进入 Past challenges 页面, 按最近完成排序后, 爬取所有相关任务的信息, 如任务标题、报酬、技术标签、发布日期等; 第 2 步, 进入每个任务页面, 在 DETAILS 页面中提取任务需求描述和任务发布日期信息; 第 3 步, 进入 REGISTRANTS 页面, 获取所有提交任务成果的开发者信息, 包括开发者名称、开发者所获名次和分数等; 第 4 步, 进入每个开发者页面, 提取有关信息. 我们进一步过滤掉信息不完整的数据. 最终, 共选择了 5 210 个任务和 1 286 个开发人员作为原始数据集.

3.2 对比算法

为充分验证本文算法的有效性, 我们设置如下两种对比算法.

(1) 面向显式特征的 FM 推荐算法 (FMRec): 该算法仅利用显式信息抽取特征, 然后利用 FM 建模不同特征和成绩的关系. 设定 FM 算法中隐因子向量的维度 f 的测试范围为{5, 10, 15, 20, 25, 30, 35, 40}, 正则化参数 λ 的测试范围均是{0.001, 0.01, 0.1, 1, 10, 100, 1000}.

(2) 面向隐式特征的 MF 推荐算法 (MFRec): 该算法仅利用成绩信息进行矩阵分解, 但是针对冷启动问题, 该算法也采用第 2.5.1 节中的多层感知器映射关系建模方法实现显式特征到隐式特征的转换. 算法中隐因子向量的维度 f 的测试范围为{5, 10, 15, 20, 25, 30, 35, 40}, 正则化参数 λ 的测试范围均是{0.001, 0.01, 0.1, 1, 10, 100, 1000}.

3.3 度量指标

我们选择 *MAE* 和 *RMSE* 两个指标评价评分预测的准确率.

$$MAE=\sum\nolimits_{(t,d,r_{td})\in T_E}|r_{td}-\hat{r}_{td}|/|T_E| \tag{12}$$

$$RMSE=\sqrt{\sum\nolimits_{(t,d,r_{td})\in T_E}(r_{td}-\hat{r}_{td})^2/|T_E|} \tag{13}$$

其中, r_{td} 和 $\hat{r}_{td}$ 分别代表任务发布者对开发者的真实评分和预测评分, T_E 是测试集数据.

此外, 我们采用准确率和召回率两个指标评价推荐结果的准确度.

$$Precision=\frac{\sum\limits_{t\in T_2}|R(t)\cap T(t)|}{\sum\limits_{t\in T_2}|R(t)|} \tag{14}$$

$$Recall = \frac{\sum_{t \in T_2} |R(t) \cap T(t)|}{\sum_{t \in T_2} |T(t)|} \tag{15}$$

其中, $R(t)$ 和 $T(t)$ 分别是预测和真实的推荐列表, T_2 表示测试的任务集.

3.4 训练与测试数据准备

3.4.1 针对冷启动任务和已有开发者情况

首先, 从评分矩阵中我们随机选择 80%, 60%, 40%, 20% 的任务和每个任务所对应开发者的数据作为训练数据, 记为 TR80, TR60, TR40, TR20. 接下来, 对剩余 20%, 40%, 60%, 80% 任务进行筛选, 保证测试数据中的开发者均为训练数据中的已有开发者, 因此, 若某个剩余任务对应的开发者是训练数据中已出现过的开发者, 则将该任务和对应开发者的数据加入测试数据, 否则则舍弃. 筛选后的测试数据记为 TE20, TE40, TE60, TE80, 用于测试 *MAE* 和 *RMSE*. 如图 4 所示任务和开发者数据包含 10 个任务和 7 个开发者, 若选择 t_1–t_8 共 8 个任务以及其关联到的 5 个开发者 d_1–d_5 为训练数据, 则任务 t_9 由于对应 2 个已有开发者 d_2、d_5 被加入到测试数据中, 而任务 t_{10} 由于对应的 2 个开发者 d_6、d_7 不在已有开发者列表中则被舍弃.

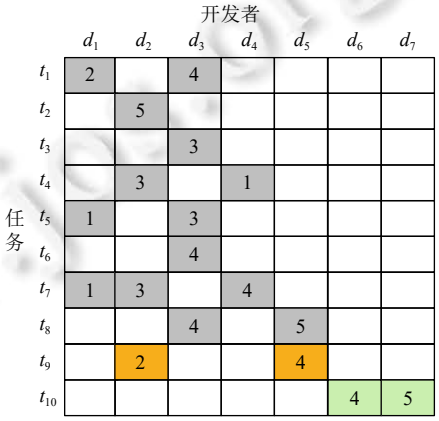


图 4 训练数据与测试数据划分示意图

需要说明的是, 为了测试准确率召回率, 对上述 TE20, TE40, TE60, TE80 需进一步处理如下: (1) 将评分大于等于 4 的标记为“推荐”, 将评分小于 4 的标记为“不推荐”; (2) 从测试任务中选择具有推荐开发者个数超过 6 个和不推荐开发者个数超过 4 个的任务, 对于该任务任意选择 6 个推荐的开发者和 4 个不推荐的开发者组成测试列表, 用于测试准确率和召回率. 实验中我们采用 10 折交叉验证进行参数调优.

3.4.2 针对冷启动任务和冷启动开发者情况

为了评价算法在包含冷启动开发者情况下的推荐性能, 首先, 采用与第 3.4.1 节相同的训练数据. 然后, 对剩余 20%, 40%, 60%, 80% 任务进行筛选, 保证测试数据中的开发者均为训练数据中从未出现过的开发者 (即冷启动开发者). 因此, 若某个剩余任务对应的开发者是训练数据中未出现过的开发者, 则将该任务和对应开发者的数据加入到测试数据, 否则则舍弃. 筛选后的测试数据也记为 TE20, TE40, TE60, TE80, 用于测试 *MAE* 和 *RMSE*. 可以发现, 此种情况下的测试数据恰好是第 3.4.1 节中舍弃的数据. 同样为了测试准确率召回率, 对上述 TE20, TE40, TE60, TE80 进行如第 3.4.1 节类似处理.

3.5 实验结果与分析

3.5.1 本文模型调优

我们基于交叉验证确定各个算法的最优参数取值, 图 5 和图 6 给出了 FM 算法和 MF 算法不同隐因子向量维度 f 和正则化参数 λ 对应的平均 *MAE* 值取值情况. 对于基于多层感知器的融合算法, 设置隐藏层层数为 1, 隐层节点个数取值范围设置为 {2, 3, 4, 5, 6}. 图 7 给出了基于多层感知器的融合算法中不同隐层节点个数对应的 *MAE* 值

取值情况. 对于基于多层感知器的显式特征到隐式特征映射建模, 由于任务的显式特征为 145 维, 我们设置多层感知器算法中第 2 层 (隐含层 1) 和第 3 层 (隐含层 2) 节点个数的取值范围为 $k_2=\{120,110,100,90,80\}$, $k_3=\{70,60,50,40,30\}$. 由于开发者的显式特征为 412 维, 我们设置 $k_2=\{220,190,160,130,100\}$, $k_3=\{70,60,50,40,30\}$. 两种情况下 k_2 和 k_3 不同参数组合的平均 *MAE* 值如图 8 和 9 所示. 图 10 给出了各个算法的迭代收敛情况. 针对本实验讨论问题 (1), 从图 5–图 9 可以看出, 本文算法的各个子模型在不同的训练集上具有不同的最优参数组合, 因此, 针对新的训练数据应该重新进行参数调优才能获取最佳的算法性能.

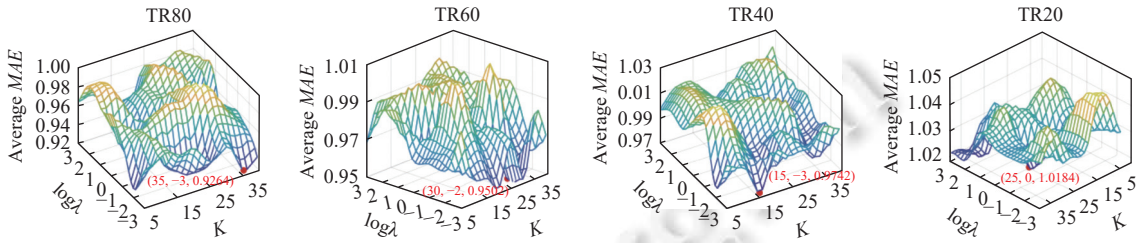


图 5 FM 算法不同参数取值对应的平均 *MAE* 值

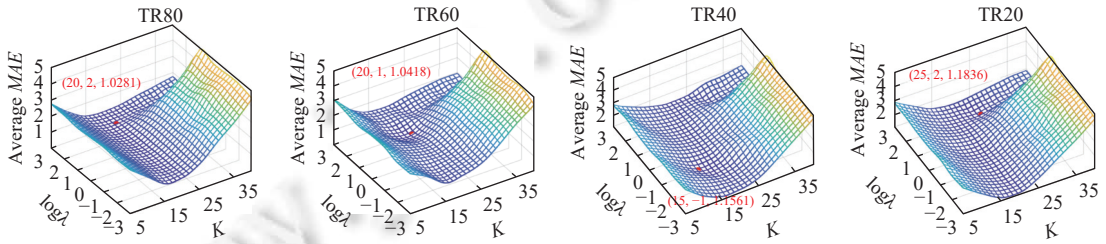


图 6 MF 算法不同参数取值对应的平均 *MAE* 值

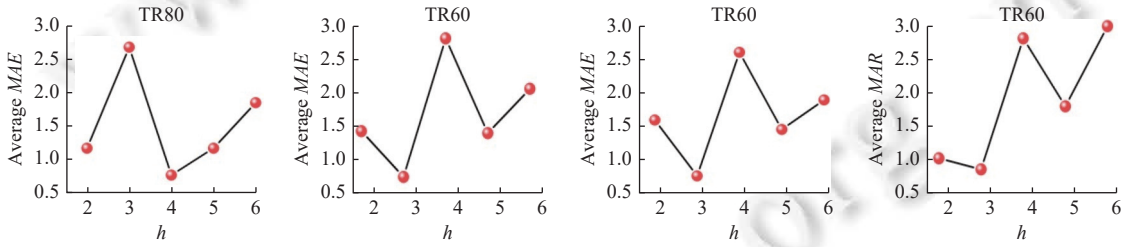


图 7 基于多层感知器的融合算法中不同隐层节点个数对应的平均 *MAE* 值

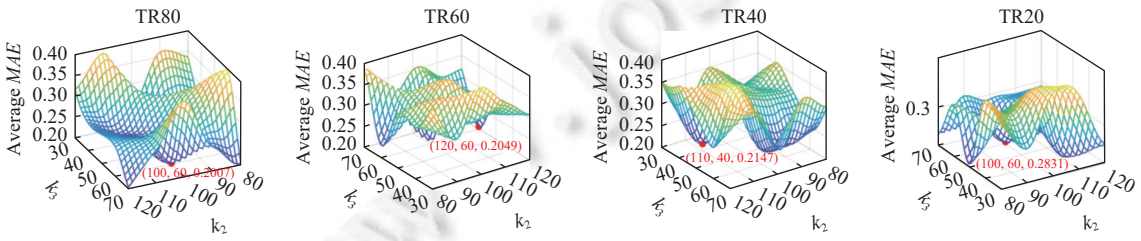


图 8 任务情况下多层感知器算法中 k_2 和 k_3 不同参数组合对应的平均 *MAE* 值

3.5.2 结果对比

本文 DHRec 算法与 FMRec 和 MFRec 算法在测试集上的 *MAE* 和 *RMSE* 对比结果如表 4 和表 5 所示, 准确率

和召回率如图 11 和图 12 所示. 其中表 4 和图 11 为冷启动任务和已有开发者情况, 表 5 和图 12 为冷启动任务和冷启动开发者的情况.

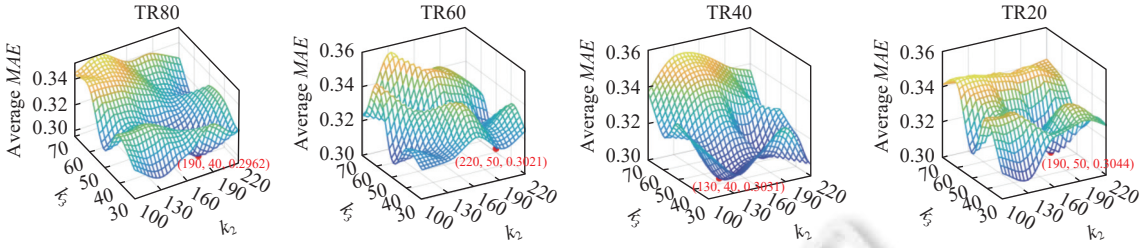


图 9 开发者情况下多层感知器算法中 k_2 和 k_3 不同参数组合对应的平均 *MAE* 值

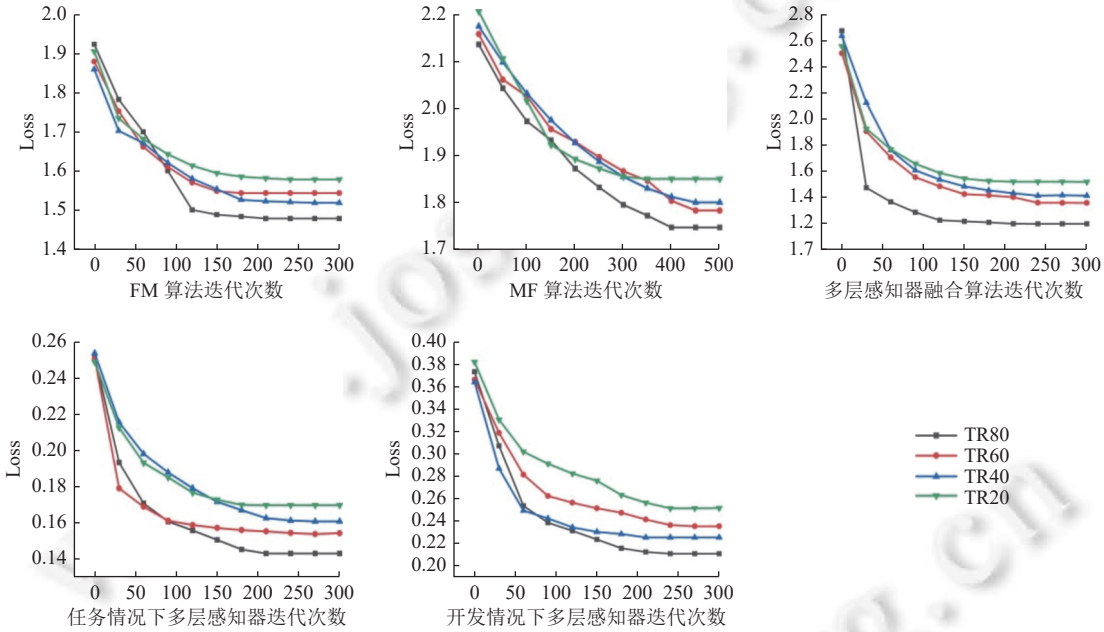


图 10 各个算法的迭代收敛情况

表 4 冷启动任务和已有开发者上 *MAE* 和 *RMSE* 对比

算法	*MAE*				*RMSE*			
	TE80	TE60	TE40	TE20	TE80	TE60	TE40	TE20
MFRec	0.9525	0.8927	0.8494	0.8154	1.2636	1.1873	1.1272	1.0763
FMRec	0.9159	0.8736	0.8315	0.8068	1.2051	1.1342	1.0934	1.0390
DHRec	0.8933	0.8648	0.8205	0.7603	1.1598	1.0827	1.0497	0.9716

从上述结果可以看出, 3 种对比算法中, 本文算法 (DHRec) 与面向显式特征的 FM 推荐算法 (FMRec) 相较于面向隐式特征的 MF 推荐算法 (MFRec) 具有一定的优势. 相对于其他两种算法, 本文方法在 4 种不同测试指标上具有明显的优势, 其中第 1 类冷启动场景下 *MAE* 值和 *RMSE* 在不同测试集上较其他方法至少平均降低 0.022 和 0.052, 准确率和召回率在不同测试集上至少平均提高 13.48% 和 11.38%. 第 2 类冷启动场景下 *MAE* 值和 *RMSE* 在不同测试集上至少平均降低 0.014 和 0.047, 准确率和召回率在不同测试集上至少平均提高 13.36% 和 11.29%. 针对本实验讨论问题 (2), 该实验的对比结果充分说明隐式特征是显式特征的有效补充, 可以缓解显式信息的不精确、含噪声困难, 通过结合显式特征与隐式特征能够获得更好的推荐性能.

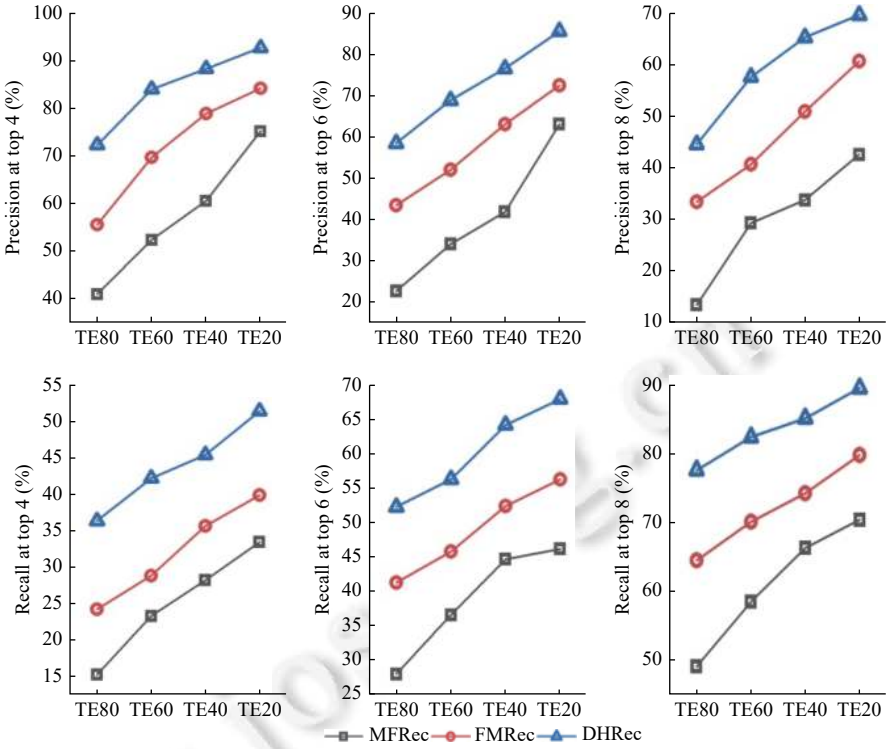


图 11 冷启动任务和已有开发者上准确率和召回率对比

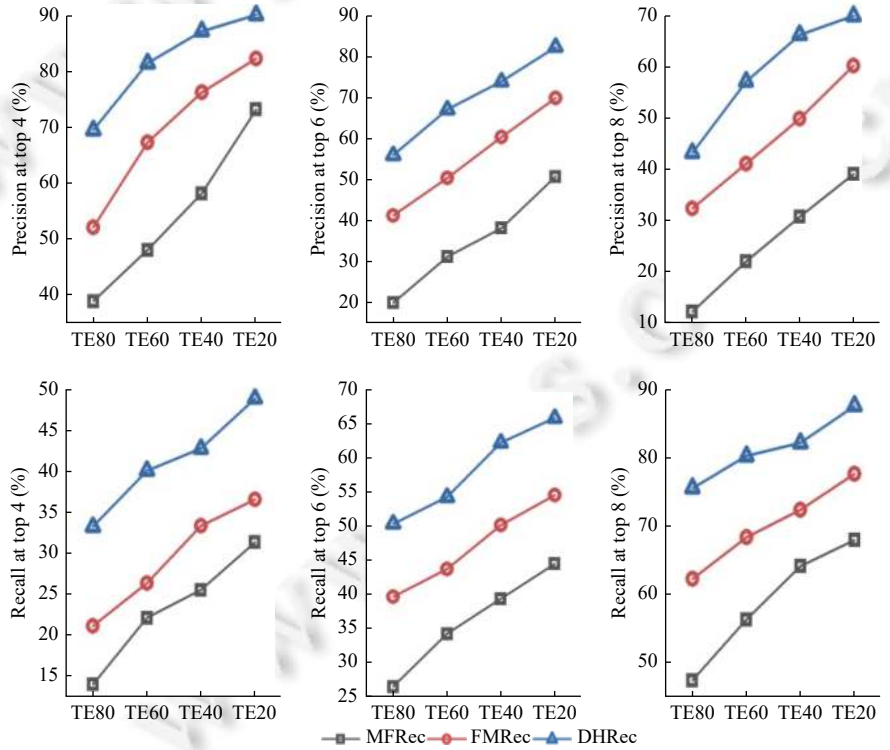


图 12 冷启动任务和冷启动开发者上准确率和召回率对比

表 5 冷启动任务和冷启动开发者上 *MAE* 和 *RMSE* 对比

算法	*MAE*				*RMSE*			
	TE80	TE60	TE40	TE20	TE80	TE60	TE40	TE20
MFRec	1.0023	0.9088	0.8603	0.8356	1.4149	1.2178	1.1351	1.1030
FMRec	0.9548	0.8807	0.8412	0.8162	1.2367	1.1442	1.1265	1.0597
DHRec	0.9437	0.8725	0.8359	0.7866	1.1854	1.1280	1.0716	0.9937

4 结 论

面向软件众包平台开发者推荐问题, 本文提出一种有效结合显式特征和隐式特征的开发者混合推荐算法, 以充分挖掘客观真实的成绩信息, 解决信息含噪声、稀疏性和推荐算法冷启动难题. 该算法分别通过 FM 和 MF 模型建模显式特征和隐式特征与成绩的映射关系, 并通过多层感知器算法构建融合模型来权衡基于 FM 的推荐模型和基于 MF 的推荐模型, 可有效解决信息含噪声、稀疏性难题. 进一步, 建模基于多层感知器的显式特征到隐式特征的映射关系, 从而可以通过冷启动任务或冷启动开发者的显式特征求解相应的隐式特征, 可有效解决冷启动难题. 在 TopCoder 平台数据集上的对比实验充分表明本文算法可以有效结合显式特征和隐式特征, 具有更好的推荐性能. 本文仅针对任务进行开发者推荐, 后续工作中我们将扩展本文模型, 实现针对开发者的任务推荐. 此外, 我们将考虑如何提高所提开发者推荐模型的可解释性.

References:

[1] Howe J. The rise of crowdsourcing. Wired, 2006, 14(6): 176–183.

[2] Mao K, Capra L, Harman M, Jia Y. A survey of the use of crowdsourcing in software engineering. Journal of Systems and Software, 2017, 126: 57–84. [doi: 10.1016/j.jss.2016.09.015]

[3] Mao K, Yang Y, Wang Q, Jia Y, Harman M. Developer recommendation for crowdsourced software development tasks. In: Proc. of the 2015 IEEE Symp. on Service-oriented System Engineering. San Francisco: IEEE, 2015. 347–356. [doi: 10.1109/SOSE.2015.46]

[4] Shao W, Wang XN, Jiao WP. A developer recommendation framework in software crowdsourcing development. In: Proc. of the 15th National Software Application Conf. Kunming: Springer, 2016. 151–164. [doi: 10.1007/978-981-10-3482-4_11]

[5] Zhu JG, Shen BJ, Hu FH. A learning to rank framework for developer recommendation in software crowdsourcing. In: Proc. of the 2015 Asia-Pacific Software Engineering Conf. New Delhi: IEEE, 2015. 285–292. [doi: 10.1109/APSEC.2015.50]

[6] Zhang ZY, Sun HL, Zhang HY. Developer recommendation for Topcoder through a meta-learning based policy model. Empirical Software Engineering, 2020, 25(1): 859–889. [doi: 10.1007/s10664-019-09755-0]

[7] Rendle S. Factorization machines with libFM. ACM Trans. on Intelligent Systems and Technology, 2012, 3(3): 57. [doi: 10.1145/2168752.2168771]

[8] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer, 2009, 42(8): 30–37. [doi: 10.1109/MC.2009.263]

[9] Balabanović M, Shoham Y. Fab: Content-based, collaborative recommendation. Communications of the ACM, 1997, 40(3): 66–72. [doi: 10.1145/245108.245124]

[10] van den Oord A, Dieleman S, Schrauwen B. Deep content-based music recommendation. In: Proc. of the 26th Int'l Conf. on Neural Information Processing Systems. Lake Tahoe: Curran Associates Inc., 2013. 2643–2651.

[11] Oppermann M, Kincaid R, Munzner T. VizCommender: Computing text-based similarity in visualization repositories for content-based recommendations. IEEE Trans. on Visualization and Computer Graphics, 2021, 27(2): 495–505. [doi: 10.1109/TVCG.2020.3030387]

[12] Fu MS, Qu H, Yi Z, Lu L, Liu YS. A novel deep learning-based collaborative filtering model for recommendation system. IEEE Trans. on Cybernetics, 2019, 49(3): 1084–1096. [doi: 10.1109/TCYB.2018.2795041]

[13] Deshpande M, Karypis G. Item-based top-*N* recommendation algorithms. ACM Trans. on Information Systems, 2004, 22(1): 143–177. [doi: 10.1145/963770.963776]

[14] Liu Z, Feng XD, Wang YC, Zuo WB. Self-paced learning enhanced neural matrix factorization for noise-aware recommendation. Knowledge-based Systems, 2021, 213: 106660. [doi: 10.1016/j.knosys.2020.106660]

[15] Wang H, Ding S, Li YQ, Li XJ, Zhang YT. Hierarchical physician recommendation via diversity-enhanced matrix factorization. ACM Trans. on Knowledge Discovery from Data, 2021, 15(1): 1. [doi: 10.1145/3418227]

[16] Lin ZQ, Chen HX. Recommendation over time: A probabilistic model of time-aware recommender systems. Science China Information

Sciences, 2019, 62(11): 212105. [doi: 10.1007/s11432-018-9915-8]

[17] Sakhi O, Bonner S, Rohde D, Vasile F. BLOB: A probabilistic model for recommendation that combines organic and bandit signals. In: Proc. of the 26th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. Virtual Event: ACM, 2020. 783–793. [doi: 10.1145/3394486.3403121] [doi: 10.1145/3394486.3403121]

[18] Bai B, Fan YS, Tan W, Zhang J. DLTSR: A deep learning framework for recommendations of long-tail Web services. IEEE Trans. on Services Computing, 2020, 13(1): 73–85. [doi: 10.1109/TSC.2017.2681666]

[19] Ni J, Huang ZH, Cheng JJ, Gao SC. An effective recommendation model based on deep representation learning. Information Sciences, 2021, 542: 324–342. [doi: 10.1016/j.ins.2020.07.038]

[20] Divyaa LR, Pervin N. Towards generating scalable personalized recommendations: Integrating social trust, social bias, and geo-spatial clustering. Decision Support Systems, 2019, 122: 113066. [doi: 10.1016/j.dss.2019.05.006]

[21] Lasmar EL, de Paula FO, Rosa RL, Abrahão JI, Rodríguez DZ. RsRS: Ridesharing recommendation system based on social networks to improve the user's QoE. IEEE Trans. on Intelligent Transportation Systems, 2019, 20(12): 4728–4740. [doi: 10.1109/TITS.2019.2945793]

[22] Lian DF, Zheng K, Ge Y, Cao LB, Chen EH, Xie X. GeoMF++: Scalable location recommendation via joint geographical modeling and matrix factorization. ACM Trans. on Information Systems, 2018, 36(3): 33. [doi: 10.1145/3182166]

[23] Wang SF, Gong MG, Wu Y, Zhang MY. Multi-objective optimization for location-based and preferences-aware recommendation. Information Sciences, 2020, 513: 614–626. [doi: 10.1016/j.ins.2019.11.028]

[24] Li CL, Quan C, Peng L, Qi YW, Deng YM, Wu LB. A capsule network for recommendation and explaining what you like and dislike. In: Proc. of the 42nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Paris: ACM, 2019. 275–284. [doi: 10.1145/3331184.3331216]

[25] Singh AP, Gordon GJ. Relational learning via collective matrix factorization. In: Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Las Vegas: ACM, 2008. 650–658. [doi: 10.1145/1401890.1401969]

[26] Hu L, Cao J, Xu GD, Cao LB, Gu ZP, Zhu C. Personalized recommendation via cross-domain triadic factorization. In: Proc. of the 22nd Int'l Conf. on World Wide Web. Rio de Janeiro: ACM, 2013. 595–606. [doi: 10.1145/2488388.2488441]

[27] Yu X, Zhan DJ, Liu L, Lv HW, Xu LW, Du JW. A privacy-preserving cross-domain healthcare wearables recommendation algorithm based on domain-dependent and domain-independent feature fusion. IEEE Journal of Biomedical and Health Informatics, 2021. [doi: 10.1109/JBHI.2021.3069629].]

[28] Pan WK, Xiang EW, Liu NN, Yang Q. Transfer learning in collaborative filtering for sparsity reduction. In: Proc. of the 24th AAAI Conf. on Artificial Intelligence. Atlanta: AAAI Press, 2010. 230–235.

[29] Yu X, Chu Y, Jiang F, Guo Y, Gong DW. SVMs classification based two-side cross domain collaborative filtering by inferring intrinsic user and item features. Knowledge-based Systems, 2018, 141: 80–91. [doi: 10.1016/j.knosys.2017.11.010]

[30] Yu X, Jiang F, Du JW, Gong DW. A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains. Pattern Recognition, 2019, 94: 96–109. [doi: 10.1016/j.patcog.2019.05.030]

[31] Li B, Yang Q, Xue XY. Can movies and books collaborate?: Cross-domain collaborative filtering for sparsity reduction. In: Proc. of the 21st Int'l Joint Conf. on Artificial Intelligence. Pasadena: Morgan Kaufmann Publishers Inc., 2009. 2052–2057. [doi: 10.1109/SIBGRAPI.2009.48]

[32] Li B, Yang Q, Xue XY. Transfer learning for collaborative filtering via a rating-matrix generative model. In: Proc. of the 26th Annual Int'l Conf. on Machine Learning. Montreal: ACM, 2009. 617–624. [doi: 10.1145/1553374.1553454]

[33] Yu X, Hu Q, Li H, Du JW, Gao J, Sun LJ. Cross-domain recommendation based on latent factor alignment. Neural Computing and Applications, 2021. [doi: 10.1007/s00521-021-05737-w]

[34] Jiang M, Cui P, Yuan NJ, Xie X, Yang SQ. Little is much: Bridging cross-platform behaviors through overlapped crowds. In: Proc. of the 30th AAAI Conf. on Artificial Intelligence. Phoenix: AAAI, 2016. 13–19.

[35] Zhang Q, Lu J, Wu DS, Zhang GQ. A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities. IEEE Trans. on Neural Networks and Learning Systems, 2019, 30(7): 1998–2012. [doi: 10.1109/TNNLS.2018.2875144]

[36] Moins T, Aloise D, Blanchard SJ. RecSeats: A hybrid convolutional neural network choice model for seat recommendations at reserved seating venues. In: Proc. of the 14th ACM Conf. on Recommender Systems. New York: ACM, 2020. 309–317. [doi: 10.1145/3383313.3412263]

[37] Tsukuda K, Goto M. DualDiv: Diversifying items and explanation styles in explainable hybrid recommendation. In: Proc. of the 13th ACM Conf. on Recommender Systems. Copenhagen: ACM, 2019. 398–402. [doi: 10.1145/3298689.3347063]

[38] Zhao XY, Zhang L, Ding ZY, Xia L, Tang JL, Yin DW. Recommendations with negative feedback via pairwise deep reinforcement learning. In: Proc. of the 24th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. London: ACM, 2018. 1040–1048. [doi: 10.1145/3219819.3219886]

[39] Zhang S, Tay Y, Yao LN, Wu B, Sun AX. DeepRec: An open-source toolkit for deep learning based recommendation. In: Proc. of the

28th Int'l Joint Conf. on Artificial Intelligence. Macao: IJCAI, 2019. 6581–6583. [doi: 10.24963/ijcai.2019/963]

[40] Xie XQ, Yang XC, Wang B, Zhang X, Ji Y, Huang ZG. Multi-feature fused software developer recommendation. Ruan Jian Xue Bao/Journal of Software, 2018, 29(8): 2306–2321 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5525.htm [doi: 10.13328/j.cnki.jos.005525]

[41] Wang ZZ, Sun HL, Fu Y, Ye LT. Recommending crowdsourced software developers in consideration of skill improvement. In: Proc. of the 32nd IEEE/ACM Int'l Conf. on Automated Software Engineering. Urbana: IEEE, 2017. 717–722. [doi: 10.1109/ASE.2017.8115682]

[42] Fu Y, Sun HL, Ye LT. Competition-aware task routing for contest based crowdsourced software development. In: Proc. of the 6th Int'l Workshop on Software Mining. Urbana: IEEE, 2017. 32–39. [doi: 10.1109/SOFTWAREMINING.2017.8100851]

[43] Ho CJ, Jabbari S, Vaughan JW. Adaptive task assignment for crowdsourced classification. In: Proc. of the 30th Int'l Conf. on Machine Learning. Atlanta: JMLR, 2013. 534–542.

[44] Shu JG, Jia XH, Yang K, Wang H. Privacy-preserving task recommendation services for crowdsourcing. IEEE Trans. on Services Computing, 2021, 14(1): 235–247. [doi: 10.1109/TSC.2018.2791601]

[45] Yuen MC, King I, Leung KS. TaskRec: A task recommendation framework in crowdsourcing systems. Neural Processing Letters, 2015, 41(2): 223–238. [doi: 10.1007/s11063-014-9343-z]

[46] Li N, Mo WK, Shen BJ. Task recommendation with developer social network in software crowdsourcing. In: Proc. of the 23rd Asia-Pacific Software Engineering Conf. Hamilton: IEEE, 2016. 9–16. [doi: 10.1109/APSEC.2016.013]

[47] Yan SH, Shen BJ, Mo WK, Li N. Transfer learning for cross-platform software crowdsourcing recommendation. In: Proc. of the 24th Asia-Pacific Software Engineering Conf. Nanjing: IEEE, 2017. 269–278. [doi: 10.1109/APSEC.2017.33]

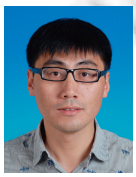
[48] Yu X, He YD, Fu Y, Xin Y, Du JW, Ni WJ. Cross-domain developer recommendation algorithm based on feature matching. In: Proc. of the 14th Conf. on Computer Supported Cooperative Work and Social Computing. Kunming: Springer, 2019. 443–457. [doi: 10.1007/978-981-15-1377-0_35]

[49] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis: Association for Computational Linguistics, 2019. 4171–4186.

[50] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proc. of the 14th Int'l Joint Conf. on Artificial Intelligence. Montreal: Morgan Kaufmann Publishers Inc., 1995. 1137–1143.

附中文参考文献:

[40] 谢新强, 杨晓春, 王斌, 张霞, 纪勇, 黄治纲. 一种多特征融合的软件开发者推荐. 软件学报, 2018, 29(8): 2306–2321. http://www.jos.org.cn/1000-9825/5525.htm [doi: 10.13328/j.cnki.jos.005525]



于旭(1982－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为推荐系统, 众包软件工程.

王昭哲(1997－), 男, 硕士生, 主要研究领域为推荐系统.

何亚东(1993－), 男, 硕士, 主要研究领域为数据挖掘与分析, 推荐系统.

江峰(1978－), 男, 博士, 教授, CCF 专业会员, 主要研究领域为机器学习, 缺陷预测.

杜军威(1974－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为众包分派, 软件测试, 形式化验证.

巩敦卫(1970－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为智能优化与控制, 基于搜索的软件工程.