软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种多特征融合的软件开发者推荐方法*

谢新强[1,2], 杨晓春[1], 王斌[1], 张霞[1,2], 纪勇[2], 黄治纲[2]

[1](东北大学 计算机科学与工程学院,辽宁 沈阳 110004)

[2](软件架构国家重点实验室(东软集团),辽宁 沈阳 110179)

通讯作者: yangxc@mail.neu.edu.cn

摘 要: 软件开发者能力评价和协作关系推荐是大数据环境下软件智能化开发领域的一个研究热点。通过分析互联网开发者社区和企业内部开发环境，设计出基于模糊综合评价的开发者能力模型；随后，通过挖掘开发者与任务的动态交互行为、静态匹配度以及开发者能力三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，最终解决开发者推荐面临的评价矩阵稀疏性和冷启动问题，提升个性化精准推荐效率。从系统层面给出适合大数据环境的多特征融合开发者推荐原型系统实践及对现有开源技术框架的优化改进，实验过程分别基于互联网问答社区 StackOverflow 和企业内部 GitLab 环境进行了实验分析。最后，对未来研究可能的问题及思路进行了展望。

关键词: 多特征融合;协作推荐;能力评价;行为感知;大数据

中图法分类号: TP311

中文引用格式: 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方法.软件学报. http://www.jos.org.cn/1000-9825/0000.htm

英文引用格式: Xie XQ, Yang XC, Wang B, Zhang X, Ji Y, Huang ZG. A Multi-Feature Fused Software Developer Recommendation. Ruan Jian Xue Bao/Journal of Software, 2017 (in Chinese). http://www.jos.org.cn/1000-9825/0000.htm

A Multi-Feature Fused Software Developer Recommendation

XIE Xie-Qiang[1, 2], YANG Xiao-Chun[1], WANG Bin[1], ZHANG Xia[1, 2], JI Yong[2], HUANG Zhi-Gang[2]

[1](School of Computer Science and Technology, Northeastern University, Shenyang 110004, China)

[2](State Key Laboratory of Software Architecture (Neusoft Corporation), Shenyang 110179, China)

Abstract: The capability evaluation and collaborative relationship recommendation of software developers is a hot topic in the field of software intelligent development in large data environment. By analyzing the Internet developer community and the enterprise internal development environment, the author designs the developer ability model based on fuzzy comprehensive evaluation. Subsequently, the three different dimensions of the dynamic interaction behavior, static matching, and developer capabilities are extracted by mining the dynamic interaction between the developer and the task, combining matrix decomposition techniques, a multi-feature fusion enhanced method based on capability and behavior for collaborative filtering developer recommendation is proposed, and ultimately solve the evaluation matrix sparseness and cold start problem of developer recommendation, enhance the personalized precision recommended efficiency. From the aspect of system, a prototype system of multi feature fusion recommendation system suitable for large data environment is presented, and the optimization of existing open source technology framework is improved. The experiment is based on the Internet Q&A community

*基金项目:国家自然科学基金(61272178, 61572122);国家重点研发计划课题“基于开发者关联分析的智能协作关键技术与支撑环境”(2016YFB1000804)

Foundation item: National Natural Science Foundation of China (61272178, 61572122);National Key Research and Development Program Key technology and supporting environment of intelligent collaboration based on relational analysis of developers (2016YFB1000804)

收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00

CNKI 在线出版时间: 0000-00-00

StackOverflow and the internal GitLab environment of the enterprise. Finally, the possible problems and ideas for future research are prospected.

Key words: multi-feature fusion; collaborative recommendation; capability evaluation; behavior perception; bigdata

近年来，随着云开发平台技术的不断成熟，软件开发正在从封闭走向开放，开发人员也逐渐从精英走向大众，通过交互分享、群体协作等方式来生产高质低价软件制品的开发模式已经取得了显著成果。开源软件社区、软件众包平台[1]、开发者社区逐渐成为开发人员解决问题的主要途径。上述趋势使得互联网平台/企业内部积累了规模巨大、异构分散、复杂演化、且价值稀疏的海量数据资源，包括开发者行为活动和相关软件代码、文档数据等。例如，短短几年时间，软件众包平台 Topcoder 就吸引了 100 多万名开发者，每天产生的资源数超过 5 万项；开发者技术社区 StackOverflow 上的用户数也达到 600 万，资源数量超过 1300 万；软件项目托管平台 GitHub 上的用户数超过 1500 万，项目数超过 3800 万；全球最大的 IT 社区 CSDN 上的用户数量也超过 3000 万，资源数量超过 9000 万。在这样一个超量信息环境下，“信息过载”问题日趋严重。面对数百万不同地域、不同经验、不同能力的开发者，想要快速准确的从中筛选出合适的开发者或开发任务是相当困难的。

在此背景下，面向软件工程领域的开发者能力评价和推荐系统的研究和应用开始受到关注。如何有效的收集、处理和分析互联网上以及企业内部的软件过程数据资源（例如互联网平台上开发者的基本信息、历史活动记录、评价信息、开发者之间或开发者与任务的关联关系信息、企业内部项目信息、源码资源、缺陷跟踪管理信息等）对开发者能力、任务分配以及开发者协作关系进行建模分析和精准推荐，从而有效地解决大数据环境下软件开发者能力评价和筛选所面临的“信息过载”问题是本文的主要研究动机和出发点。

本文第 1 章介绍相关工作。第 2 章提出基于模糊综合评价的开发者能力评价模型。第 3 章提出多特征融合的协同过滤开发者推荐方法以及大数据环境下开发者推荐系统实践的技术方案选型和优化过程。第 4 章为实验分析。第 5 章给出结论及未来工作展望。

1 相关工作

推荐系统作为解决“信息过载”问题的有效手段，其概念是由卡内基•梅隆大学 Robert Armstrong 在 1995 年 AAAI 会议上首次提出，并推出第一个推荐系统的原型 Web Watcher。随后，斯坦福大学 Marko Balabanovic 等人推出的个性化推荐系统 LIRA1 为个性化推荐系统的发展奠定了基础[2]。随着电子商务、社交网络的异军突起，学术界和产业界对于推荐系统的研究和应用也异常火爆。其中，协同过滤推荐算法（Collaborative Filtering，CF）是推荐系统领域最经典的和最流行的算法，例如 Amazon 电子商务推荐系统、腾讯视频推荐系统、豆瓣音乐推荐系统等都采用了协同过滤推荐方法。协同过滤推荐方法的优点在于不依赖物品内容，推荐结果具有多样性，缺点也非常明显，新用户和新物品的冷启动问题、评价矩阵的数据稀疏问题一直是其无法逾越的瓶颈问题。例如，Netflix 百万大赛中提供的音乐数据集的稀疏度是 1.2%，而淘宝网的评价矩阵数据稀疏度甚至仅在 0.01%左右。对这一问题，学术界和产业界进行过大量的探索，也提出了许多有价值的研究思路。如文献[3]、[4]在协同过滤评价矩阵基础上加入了时间因子、物品类型、人口统计学特征；文献[5]-[6]借助社交媒体、用户共享社区和 Tweet 上蕴含的 Topic 获取用户潜在的偏好特征，并融合社会网络、内容相关性、时间、地理位置等因素特征；文献[7]、[8]利用网页浏览、检索和 Tweet 上广告点击等用户行为日志数据来改善评价矩阵的数据稀疏问题；文献[9]、[10]借助不同用户兴趣偏好的地域性和朋友圈分享内容的关联性特征来改善数据稀疏和冷启动问题；文献[11]-[13]基于 OCCF(One Class Collaborative Filtering)方法对评价矩阵采用“零值注入”的方法来解决评价矩阵的数据稀疏和冷启动问题等。

近十年，尽管推荐系统在电子商务、社交媒体、新闻广告、搜索引擎等领域的研究和应用已经取得了很大的成功。但对于软件工程领域的资源获取、任务分配、开发者、协作关系等方面的推荐系统（Recommendation systems for software engineering，RSSEs）成功案例却非常少[14]。一方面由于传统软件开发主要基于开发者本

地或者企业内部的封闭环境下进行，这使得对于异构、多源、碎片化分布的软件工程大数据的自动化收集相当困难。另一方面，由于软件开发是知识密集型活动，个体能力差异和社会化生产环节是影响效率的重要因素，但这些因素又都没有一致的度量和评价标准，导致软件开发过程中所涉及的开发者能力评价和任务推荐的工具和系统很难落地[15]。以互联网开发模式为例，软件项目主要是通过开源社区或众包平台形式对外发布，而开发者通过自组织协作和竞争的方式获得项目开发机会，研究发现，开发者与项目之间的连接矩阵非常稀疏，只有少数开发者与少数项目之间存在明显评价关系。这导致了现有开发者推荐方法效果极差。为解决这一问题，学术界进行了大量研究，如文献[16]基于 KNN 方法分析缺陷报告中潜在的 Topic 和开发者特征之间的相似关系来推荐最佳的缺陷修复者；文献[17]提取多个域中的 Topic，计算跨域 Topic 间的相似关系进行跨域协作关系推荐；文献[18]基于 Topcoder 上历史获胜者、参与者、任务特征信息，提取出开发者与任务的关联关系，实现为特定项目推荐开发者等。文献[16]-[18]是从静态特征挖掘开发者和 Topic 的相似关系，实现为特定 Topic 推荐开发者，但并没有考虑到开发者的动态行为特征，而实际上由于开发者与 Topic 的关系矩阵的稀疏问题，使得其预测结果的平均准确度和覆盖率并不高。文献[19]通过分析 GitHub 上用户对特定技术术语的使用频度，问答社区（StackOverflow）上标签和技术术语的关联关系，实现为 StackOverflow 推荐问答专家。文献[20]对 StackOverflow 上开发者的历史数据分析，并基于 LDA 主题模型分析和发现用户潜在的兴趣，最终基于用户兴趣和协作投票机制相结合的方式为 StackOverflow 推荐问题专家。文献[21]基于开发者历史完成任务和开发者声誉结合的方式，给出开发任务匹配和推荐模型，利用开发者的声誉增强推荐结果的准确度和多样性。其缺点是基于内容推荐的方式需要对每个任务进行特征抽取，计算过程复杂且推荐结果的新颖性较差，此外，由于大量新加入或者已加入的开发者由于没有参与过任务，将很难得到被推荐的机会，使得冷启动问题依然存在。

综上所述，现有工作难点和不足主要概括如下两点：

(1) 开发者推荐的难点是解决评价矩阵的数据稀疏和冷启动问题，通常当评价矩阵过于稀疏或冷启动状态下，推荐结果的准确度和覆盖率等方面的效果极差，而这一问题目前尚没有很好的解决方法；

(2) 现有方法的不足之处在于单纯考虑开发者和任务的评价矩阵，没有考虑开发者的“显式”能力特征，使得推荐结果的可解释性不强；另外，缺乏对开发者“隐式”动态行为特征的挖掘，实际上开发者与任务之间存在大量的交互行为，这对解决推荐系统评价矩阵数据稀疏和冷启动问题是非常有价值的。

针对现有工作存在的问题和不足，本文提出了一种基于开发者能力与行为多特征融合的推荐方法，其主要贡献包括如下三点：

(1) 提出一种基于模糊综合评价的开发者能力评估模型，并给出了开发者能力模型、开发者与任务关联关系的形式化描述；

(2) 提出一种基于开发者能力和行为感知的多特征融合开发者推荐方法并给出面向大数据环境下的开发者实时推荐系统的开发和实践过程描述；

(3) 通过对互联网问答社区 StackOverflow 和 IT 企业内部 GitLab 源码管理的真实数据分析给出了本文方法和系统的有效性证明。

Table 1 Relative symbol definition

表 1 相关符号定义描述

符号	描述	符号	描述
U	开发者集合	C	开发者能力特征集合
V	评分等级集合	W	能力特征权重集合
r_{ij}	能力特征 c_i 在评价等级 v_j 上的隶属度	X(u)	开发者 u 能力评价向量
$dcm(u)$	开发者能力模型	UT	开发者 u 特征标签集合
KT	任务 t 的特征标签集合	A	开发者 u 与任务 t 交互项集合

$v(u,t,a_k)$	开发者 u 对任务 t 关于交互项 a_k 的执行次数	$ad(u,t)$	开发者 u 与任务 t 的关联度
$md(u,t)$	开发者 u 与任务 t 的匹配度	α_k	开发者 u 对任务 t 的交互项 a_k 的权重，$\alpha_k \in \alpha$
R^{init}	初始评价矩阵	P_t^{T}	任务 t 对应的 d 维隐因子向量
R^{ad}	加入 $ad(u,t)$后增强的评价矩阵	Q_u	开发者 u 对应的隐因子向量
R^{mf}	矩阵分解拟合的评价矩阵	σ	$r^{init}{}_{tu}$与 $ad(u,t)$权重因子
R^{mfd}	多特征融合优化后的评价矩阵	λ	正则化权重系数
$r^{init}{}_{tu}$	$r^{init}{}_{tu} \in \mathrm{R}^{init}$ 初始评价值	b_u	开发者 u 的评价偏差
$r^{ad}{}_{tu}$	$r^{ad}{}_{tu} \in \mathrm{R}^{ad}$ 加入 $ad(u,t)$增强后的评价值	b_t	任务 t 的评价偏差
$r^{mf}{}_{tu}$	$r^{mf}{}_{tu} \in \mathrm{R}^{mf}$ 矩阵分解拟合的评价值	$\bar{r}$	评价矩阵各元素均值
$r^{mfd}{}_{tu}$	$r^{mfd}{}_{tu} \in \mathrm{R}^{mfd}$ 多特征融合优化后的评价值	$\lambda_1,\lambda_2,\lambda_3$	多特征融合权重
η_0	学习速率因子η的初始值	$tsd(i,j)$	任务 i、j 的相似性

2 开发者能力特征模型

开发者能力评价问题是开发者推荐系统需要解决的难点问题。对于开发者能力的建模和评价标准，学术界和产业界至今没有一致的定义。这导致了软件协作开发过程中由于缺少对开发者能力、技术、经验、影响力等特征的合理度量和评价标准，使得对于开发者推荐结果的可信性往往缺乏说服力和可解释性。对于 IT 企业而言，主要依靠项目经理的主观经验去选择合适的开发者作为任务的执行者。而基于互联网平台的软件开发模式中，网络上分布着上百万的开发者，每个开发者的能力、技术和经验良莠不齐、难以甄别。在这种情况下，很难根据主观经验对每一个开发者度量、评价和筛选。因此，本文开发者推荐首要解决的问题是建立开发者能力度量和评价模型，对开发者能力特征从不同维度特征、以定性分析和定量分析相结合的方式进行的抽象刻画。例如，对开发者特征（如技能、学历、经验、活跃度、关注度、以及开发者人口统计学特征等）的抽象。

2.1 开发者能力模型定义

开发者个体能力的评价问题是一个涉及多学科融合、层次交叠、不断演化的复杂性系统问题，需要综合、全面的考察开发者的技能经验、认知能力、教育背景、社区活跃度和影响力等多个维度的特征。现有研究通常仅对开发者某个维度的特征进行建模和分析，比如单纯的基于评价矩阵、开发者社会关系网络、标签、历史操作记录等，缺少对各因素的综合考虑，导致评价结果容易受单一因素的影响而产生较大的偏差。本文将模糊综合评价的基本思想[22]引入到开发者能力评价的计算过程中，通过分析开发者静态和动态、显式和隐式特征，以定性和定量分析相结合的方式，给出开发者能力模型的模糊综合评价方法。模糊综合评价是建立在模糊集合基础之上，运用模糊数学原理对受多种因素影响的事物做出全面、客观评价的一种决策方法。定义 1 为开发者能力模糊综合评价模型的定义。

定义 1：开发者能力模型

设 $\mathrm{U}=\{u_1,u_2,\ldots,u_n\}$表示开发者集合，有限论域 $\mathrm{C}=\{c_1,c_2,\ldots,c_n\}$为开发者的能力特征集合，$\mathrm{V}=\{v_1,v_2,\ldots,v_n\}$为开发者能力特征对应的评分等级集合，$\mathrm{W}=\{w_1,w_2,\ldots,w_n\}$为开发者能力特征对应的权重集合，且满足 $v_i\in[0,1]$，$w_i\in[0,1]$，$\sum_{i=1}^{n} w_i = 1$，$r_{ij}\in[0,1]$表示评价特征 c_i 在评价等级 v_j 上的模糊隶属度，则开发者 u 的能力评价向量定义为：

$$\mathrm{X}(u)=\left(\sum_{i=1}^{n} x_i\right)^{-1}\cdot[x_1,x_2,\ldots,x_n]^{\mathrm{T}} \qquad (1)$$

其中，$x_i=\sum_{j=1}^{m} r_{ij}\cdot v_j$，$\left(\sum_{i=1}^{n} x_i\right)^{-1}$是归一化处理。根据模糊综合评价方法，开发者能力模型（Developer

Competency Model）的形式化描述为：

$$dcm(u)=\left(\sum_{i=1}^{n}x_i\right)^{-1}\cdot\sum_{i=1}^{n}w_i\cdot x_i \qquad (2)$$

$dcm(u)\in[0,1]$是对开发者能力从多个特征维度的模糊加权综合计算结果。例如，假设开发者 u 的能力特征为 C={ c_1：“社区活跃度”，c_2：“社区贡献度”， c_3：“社区影响力”，c_4：“项目经验”，c_5：“语言能力”}，对应的评分等级 V={ v_1：0.3，v_2：0.6，v_3：0.9 }，能力特征对应的权重 W={ w_1：0.2，w_2：0.2，w_3：0.3，w_4：0.2，w_5：0.1}，开发者能力特征对应等级的模糊隶属度矩阵如表 2 所示，本文示例中隶属度是基于经验值给出。

Table 2 Developer competency characterization sample

表 2 开发者能力特征描述示例

	c_1	c_2	c_3	c_4	c_5
v_1	0.1	0.2	0.3	0.4	0.5
v_2	0.4	0.2	0.3	0.3	0.4
v_3	0.5	0.6	0.4	0.3	0.1

则根据公式 1 和公式 2 的定义，开发者能力评价向量 $X(u)=[0.22,0.22,0.23,0.18,0.15]^T$，开发者能力模糊评价计算结果为 $dcm(u)=0.065$。定义 1 基于模糊理论给出了开发者能力向量和模糊综合能力评价的计算结果，充分考虑了可能的影响因素和权重，相比单纯评价指标而言，其计算结果更具全面性和客观性。

2.2 开发者及任务关系

本文将开发者与任务之间的关系划分为开发者-任务静态匹配关系和开发者-任务动态关联关系。开发者-任务匹配关系是基于标签相似性计算获得，是开发者与任务之间静态特征关系的量化。开发者-任务关联关系则是对开发者交互行为形成的与任务之间的关联关系，是对开发者动态特征关系的量化，比如开发者通过访问、提交、评论、下载等交互操作与任务之间建立的关联关系。如下定义 2、定义 3 分别给出了匹配关系和交互关系的量化，本文称为“匹配度”、“关联度”。

定义 2：匹配度

假设开发者 u 的标签集合为 $UT=\{ut_1,ut_2,\ldots ut_n\}$,任务 t 的标签集合为 $KT=\{kt_1,kt_2,\ldots kt_m\}$。则根据 Jaccard 相似性系数[7]原理，开发者 u 与任务 t 的匹配度（Matching Degree）定义如下：

$$md(u,t)=\frac{|UT\cap KT|}{|UT\cup KT|} \qquad (3)$$

$md(u,t)\in[0,1]$表示开发者与任务的相似匹配程度，当 $md(u,t)$越大，匹配度越高。例如，StackOverflow 网站上的一个真实的开发者标签 UT={“android”,“html”,“angular”,“css”,“java”}，任务标签为 KT={“javascript”,“html”,“angular”,“web-hosting”}，则根据公式（3）的定义，其匹配度为 $2/7\approx0.29$。基于标签的相似度计算和推荐算法是目前推荐系统中比较流行的使用方法，但标签自身存在书写规范性或异词同义问题，通常数据预处理阶段需要对标签进行必要的数据清洗或分词处理。

定义 3：关联度

假设 $A=\{a_1, a_2,..., a_n\}$表示开发者 u 对任务 t 的 n 个交互项集合，$v(u,t,a_k)$表示开发者 u 对任务 t 关于交互项（或称为动作）a_k 的执行次数，则开发者 u 与任务 t 的关联度（Association Degree）的形式化定义如下：

$$ad(u,t)=\sum_{k=1}^{n}\alpha_k\cdot\frac{v(u,t,a_k)}{\sum_{u\in U}v(u,t,a_k)} \qquad (4)$$

其中，$ad(u,t)\in[0,1]$，$\alpha_k\in[0,1]$表示任务 t 的交互项 a_k 的权重因子，且 $\sum_{i=1}^{n}\alpha_i=1$，关联度的定义是对开发者与任务关联关系的定量化分析，是从开发者与任务的动态交互行为分析的角度描述开发者与任务关联关系的强弱程度。其中，任务 t 可看作开发者执行的目标对象，如互联网技术社区、开源社区、软件众包平台上的开发项目、开发任务、缺陷、帖子、问答、评论等。而动态交互项则可看作是对项目、任务、缺陷、帖子、问

答、评论等执行、访问、提交、下载、回答、点赞、分享、‘@’等一系列动作集合。

3 多特征融合的开发者推荐方法及实践

为解决协同过滤开发者推荐系统实践过程中存在的评价矩阵稀疏和冷启动问题，本文提出一种能力与行为多特征融合的协同过滤开发者推荐方法，基本思想是通过分析开发者动态行为特征并利用矩阵分解拟合技术对评价矩阵进行增强优化，通过对增强后的评价矩阵、开发者能力特征和开发者-任务相似匹配度特征融合，实现多特征融合的协同过滤开发者推荐方法。实现原理如图 1 所示，序号①是为解决评价矩阵的数据稀疏性问题，首先通过开发者与任务的交互关联关系对稀疏的原始评价矩阵 R^{init} 进行增强优化，生成增强优化后的矩阵 R^{ad}。其次，如②所示，为通过矩阵分解拟合来预测评价矩阵中的未知项，对 R^{ad} 进行矩阵分解生成拟合后的评价矩阵 R^{mf}。然后，如③所示，为提升开发者推荐结果的准确性、相关性、可解释性问题，通过对增强后的开发者评价矩阵、开发者-任务相似匹配关系、开发者能力多个特征进行加权融合来构造多特征融合后的评价矩阵 R^{mfd}。根据融合后的评价 R^{mfd} 查询相似任务，构造相似任务集合，最后基于协同过滤推荐算法，生成 TOP-K 推荐列表。

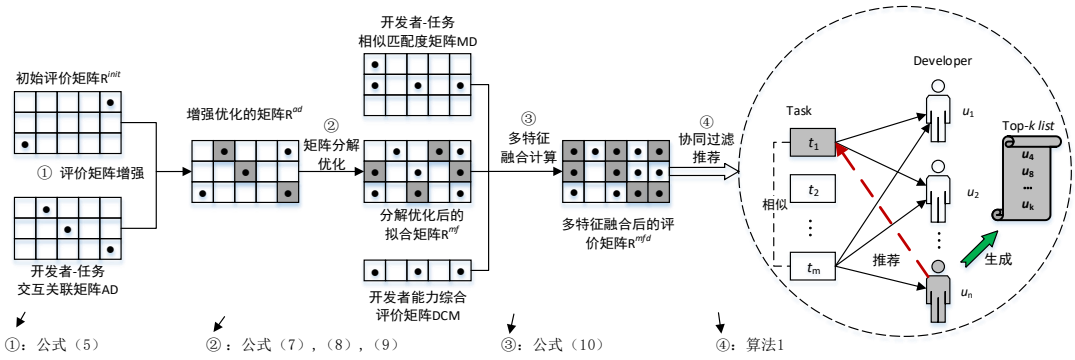


Fig.1 Diagrammatic sketch of multi-feature fusion CF developer recommendation

图 1 多特征融合的协同过滤开发者推荐方法示意图

3.1 节为基于开发者行为感知的评价矩阵增强优化方法，3.2 节为评价矩阵分解优化方法，3.3 节为协同过滤的开发者推荐生成算法，3.4 节为大数据环境下多特征融合推荐系统实践。

3.1 基于开发者行为感知的评价矩阵增强方法

令 R^{init} 表示任务-开发者初始评价矩阵，$r^{init}_{tu} \in R^{init}$ 表示针对任务 t，开发者 u 获得的初始评价值，r^{init}_{tu} 描述的是开发者关于任务获得的“显式”评价。由于初始评价矩阵 R^{init} 存在数据稀疏问题，甚至很多开发者或技术社区不存在 “显式”的评价矩阵。为解决这一问题，本节提出基于开发者行为感知的开发者评价矩阵增强方法，其核心思想是为任务推荐开发者时，不仅考虑开发者与任务的“显式”评价关系，同时综合考虑开发者与任务的“隐式”交互行为关系，对 r^{init}_{tu} 增强优化的形式化描述如下：

$$r^{ad}_{tu} \leftarrow \sigma \cdot r^{init}_{tu} + (1-\sigma) \cdot ad(u,t) \qquad (5)$$

$\sigma \in [0,1]$为权重因子，$r^{ad}_{tu} \in R^{ad}$ 表示增强后评价值，$ad(u,t)$（见公式（4）），为开发者与任务的动态交互关联关系。初始评价矩阵是对开发者“显式”评价特征的描述，而开发者与任务的交互关系则是开发者与任务之间行为特征的刻画，是“隐式”特征的量化。公式（5）通过引入开发者行为特征来增强评价矩阵，能够有效增强评价矩阵中的已知项数量，解决评价矩阵数据稀疏所造成的推荐精确度低及冷启动问题。图 2 给出了任务-开发者评价矩阵增强的示意图，白色圆点格子表示初始评矩阵中的已知项，灰色圆点格子是加入开发者动态交互行为 $ad(u,t)$对评价矩阵增强后产生的已知项。

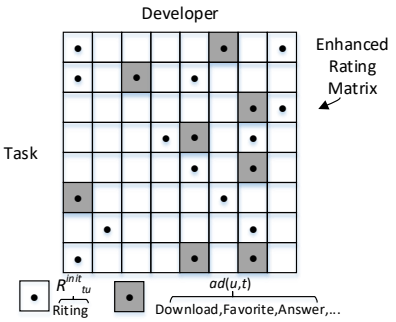


Fig.2 Diagrammatic sketch of rating matrix enhancement

图 2　评价矩阵增强示意图

3.2 基于矩阵分解的评价矩阵优化方法

为提升推荐系统预测的准确性，同时降低大数据环境下评价矩阵因高维、稀疏所导致的计算复杂性问题，本节基于矩阵分解(Maxtrix Factorization,MF)技术[3]，对 3.1 节增强后的开发者评价矩阵 R′进行分解和拟合优化，以便更好的预测出评价矩阵中存在的未知项，矩阵分解方法在解决大数据环境下评价矩阵的数据稀疏、高维等问题方面相比 SVD、近邻模型和受限波兹曼机等方法，不仅计算简单而且可用性好[8]。

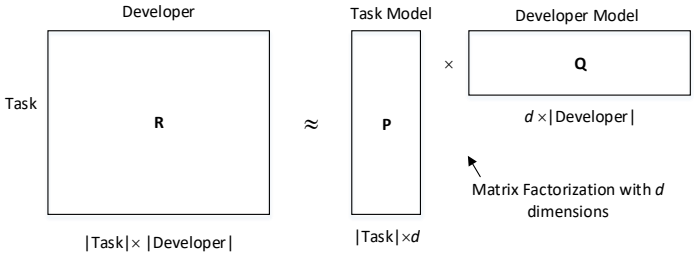


Fig.3 Diagrammatic sketch of rating matrix factorization

图 3　评价矩阵分解示意图

如图 3 所示，其核心思想是将开发者能力特征和任务特征投影到同一个低维的隐因子空间（Latent Factor Space）Γ^d 中，d 表示空间维度，假设每个开发者 u 对应一个隐因子向量 $Q_u \in \Gamma^d$，每个任务 t 对应一个隐因子向量 $P_t \in \Gamma^d$，则开发者 u 对于任务 t 的评价关系近似为 $P_t^{\mathrm{T}} \cdot Q_u$，其中 Q_u 中每个元素代表开发者 u 在对应的隐因子上的隶属度。同理，P_t 中的每个元素表示任务 t 在对应隐因子上的隶属度。开发者评价矩阵分解的目的是求解向量 P_t^{T} 和 Q_u 的内积，假设 $r^{mf}_{tu} \in \mathrm{R}^{mf}$ 表示矩阵分解拟合后的评价值，则 $r^{mf}_{tu}=P_t^{\mathrm{T}} \cdot Q_u$，。为求解 $P_t^{\mathrm{T}} \cdot Q_u$，采用最小化平方误差法定义损失函数，假设 D 为评价矩阵 R 中的已知项集合，则最小化损失函数定义如下：

$$\min_{t^*,u^*} \sum_{(t,u)\in D} \left(r^{ad}_{tu} - P_t^{\mathrm{T}} \cdot Q_u\right)^2 + \lambda\left(\|P_t\|^2 + \|Q_u\|^2\right) \quad (6)$$

其中，$\|\ \|$是对向量求范数，λ ($\|P_t\|^2+\|Q_u\|^2$)是正则化因子，目的是防止最小化损失函数求解存在过拟合（overfitting）问题。$\lambda \in [0,1]$为实验调节因子。但公式（6）并未考虑实际应用中，可能存在的评价偏差。例如，有些开发者偏向评价低分（过于苛刻）；有些任务的评价偏高（故意炒作或人为作弊）等。显然，评价矩阵具有一定的主观性，容易受到不确定因素的影响而产生偏差，这种偏差影响了整体评价结果的可信性。针对这一问题，本文提出在公式（6）的基础上进行偏差修正处理，给出带偏置项的矩阵分解方法如下：

$$\min_{t^*,u^*} \sum_{(t,u)\in D} \left(r^{ad}_{tu} - \bar{r} - b_u - b_t - P_t^{\mathrm{T}} \cdot Q_u\right)^2 + \lambda\left(\|P_t\|^2 + \|Q_u\|^2 + {b_u}^2 + {b_t}^2\right) \quad (7)$$

其中，$\bar{r}$ 为评价矩阵 R^{ad} 中元素的平均值，b_u 是开发者 u 的评价偏差，b_t 是任务 t 的评价偏差。对公式（7）采用随机梯度下降法(Stochastic Gradient Descent, SGD) [4]求解，SGD 与交替最小二乘法相比，在解决数据稀疏问题、模型精度、运算速度方面具备明显优势，适合于解决大数据量的最优化求解问题。基于随机梯度下降法对公式（7）进行求解得到的参数更新过程如公式（8）所示：

$$\begin{aligned} b_u &\leftarrow b_u + \eta\left(e_{ut} - \lambda b_u\right) \\ b_t &\leftarrow b_t + \eta\left(e_{ut} - \lambda b_t\right) \\ Q_u &\leftarrow Q_u + \eta\left(e_{ut} \cdot Q_u - \lambda Q_u\right) \\ P_t &\leftarrow P_t + \eta\left(e_{ut} \cdot P_t - \lambda P_t\right) \end{aligned} \tag{8}$$

其中，$e_{ut} = r^{mf}{}_{tu} - \bar{r} - b_u - b_t - P_t^{\mathrm{T}} \cdot Q_u$ 为评价误差。η是学习速率调节因子。根据公式（8）的计算结果，求出预测的评价矩阵为 R^{mf}，其对应元素的计算方法如下：

$$r^{mf}{}_{tu} = r^{ad}{}_{tu} - \bar{r} - b_u - b_t - P_t^{\mathrm{T}} \cdot Q_u \tag{9}$$

公式（9）给出了分解拟合后的评价矩阵，目的是借助矩阵分解和再拟合过程来预测评价矩阵中存在的未知项，从而进一步解决评价矩阵的数据稀疏和冷启动问题，提高开发者推荐系统的预测准确性。

3.3 多特征融合的协同过滤开发者推荐方法

推荐系统领域的特征融合主要包括模型融合和推荐结果融合两种方式[23]。模型融合是相对于推荐结果融合而言，是将多个特征模型作为输入，输出融合后的特征模型，在此模型基础上进行推荐结果的计算。而推荐结果融合则是基于多个特征模型的单独计算，在对各自的推荐结果按不同策略进行融合的方式。基于这一思想，在公式（9）的基础上，进一步融合开发者能力、任务-开发者的匹配度静态特征，基于多特征融合的方式提升协同过滤推荐结果的准确性、相关性和可解释性。令 R^{mfd} 表示多特征融合优化后的评价矩阵，$r^{mfd}{}_{tu} \in R^{mfd}$ 表示开发者融合推荐值，则其计算过程如下：

$$r^{mfd}{}_{tu} = \lambda_1 \cdot r^{mf}{}_{tu} + \lambda_2 \cdot dcm(u) + \lambda_3 \cdot md\left(u,t\right) \tag{10}$$

其中，$r^{mfd}{}_{tu}$、$dcm(u)$、$md(u,t)$分别由公式（9）、（2）、（3）定义。λ_1、λ_2、$\lambda_3 \in [0,1]$为权重因子，且满足条件$\lambda_1+\lambda_2+\lambda_3=1$。公式（10）基于多特征推荐融合给出了开发者综合评价结果，从多个特征维度进行了综合考虑，即降低了评价矩阵稀疏性问题、冷启动问题对推荐结果的影响，同时又从不同维度对开发者能力进行了整体度量，提高了推荐结果的可信性和可解释性。而且，更重要的是通过对λ_1、λ_2、λ_3权重的动态调整，能够满足不同企业、技术社区、软件众包平台对开发者能力特征的关注重点。例如，对于大型 IT 企业而言，更侧重于开发者的个人能力， 则λ_2可以给予较大权重；对于中小型企业而言，由于受到企业规模和成本的限制，更关注开发者技能匹配度和业务熟练程度，则λ_3 可以给予较大权重，而对于初创企业，比较看重开发者的新技术关注度、社区活跃度等，则λ_1 的权重设置较大。

基于评价矩阵 R^{mfd} 给出多特征融合增强的协同过滤开发者推荐（Multi-Feature Fused Developer Recommendation, MFD_Rec）。如算法 1 所示，首先，根据公式（11）计算任务之间的相似度 *tsd*（Task Similarity Degree）。然后，对每个任务 i，找到与其最相似的 s 个任务集合 $T_i=\{t_1,t_2,\ldots,t_s\}$，查询 T_i 中任务关联的开发者并排除任务 i 已拥有的开发者，形成对应开发者集合 $U=\{u_1,u_2,..u_x\}$，对 U 中每个开发者，计算 $tsd(i,j) \cdot r^{mfd}{}_{ju}$，将计算结果存储在 *list* 中，对 *list* 排序，并返回 *list* 的 top-k 个元素所对应的开发者。

如公式（11）所示，对任务相似性计算，由于余弦相似度计算方法更多的是从方向上区分差异，而对绝对的数值不敏感，因此难以衡量每个维度上数值的差异。基于这一问题，本文给出了基于修正的余弦相似度算法(Adjusted Cosine Similarity)[3]来求解任务之间的相似性，其优点在于考虑了不同开发者的评分尺度问题，通过减去每个维度上的平均值，降低了相似度计算的误差。另外，由于相似度矩阵是对称矩阵，所以算法 1 中借助一维数组 a 对相似度矩阵进行了压缩存储优化，如算法 1 中的序号 3 描述。

$$tsd(i,j)=\frac{\sum_{u=1}^{n}\left(r_{iu}-\overline{r_i}\right)\cdot\left(r_{ju}-\overline{r_j}\right)}{\sqrt{\sum_{u=1}^{n}\left(r_{iu}-\overline{r_i}\right)^2}\cdot\sqrt{\sum_{u=1}^{n}\left(r_{ju}-\overline{r_j}\right)^2}} \quad (11)$$

多特征融合增强的协同过滤开发者推荐算法伪代码描述如下：

Table3 Multi-feature fused developer recommendation generation

表 3 多特征融合的协同过滤开发者推荐生成

算法 1. Multi-Feature Fused Developer Recommendation,MFD_Rec.

Input：R^{mfd}, k; /* R^{mfd} is fused rating maxtrix, k is the number of result*/
Output：top-k developers /*the result list */
1:for i=0 to n-1 /*i, j is the index of task, n is the number of task */
2: for j = i+1 to n /* init task similarity degree by (11) */
3: $a[i\cdot(i+j)/2+j] \leftarrow tsd(i,j)$; /* a is an array to store compressed maxtrix */
4: end for
5:end for
6:for i =0 to n-1
7: $T_i[0\ldots s\text{-}1]\leftarrow$ get i's top-s neighbors based array a ;
8: for j = 0 to s-1 /*get neighbors' developers */
9: U[0…x]← get developers related to $T_i[j]$ && developers not related to i ;
10: for u in U[0…x]
11: $list\,[u]\leftarrow tsd(i, T_i[j])\cdot r^{mfd}{}_{Ti[j]\,u}$; /* $tsd(i, T_i[j])$ by(11), $r^{mfd}{}_{Ti[j]\,u}$ by (10)*/
12: end for
13: end for
14:end for
15:sort(*list*); /*sort the list order by list element desc*/
16:return top-k developers;

算法 1 是在评价矩阵增强、分解拟合、多特征融合优化的基础上给出的协同过滤推荐方法，从评价矩阵增强的基础上提出了基于多特征融合的协同过滤开发者推荐方法实现。

3.4 大数据环境下开发者推荐系统实践

本节在多特征融合开发者推荐算法基础上，实现了面向大数据环境的开发者实时推荐系统，包括数据实时采集、预处理、索引、存储、特征提取和实时计算过程。系统基于 ElasticSearch(以下简称“ES”)、Spark 等目前业界最流行的开源技术框架实现，能够满足大数据环境下开发者推荐系统的实时计算、大并发、水平伸缩性和可用性要求。如图 4 所示，为系统整体实现框架，核心部件包括数据采集工具：爬虫工具 Crawler 、日志数据收集工具 Logstash、网络抓包工具 PackageBeta；数据预处理和索引工具 Logstash；数据存储检索框架 ES；计算框架 Spark 等。系统支持处理 PB 级结构化或非结构化数据，具备秒级的查询处理延迟，支持对互联网技术社区页面数据抓取和企业内部 GitLab 数据的实时采集。对企业内部数据支持通过数据监听 Agent（Logstash、PackageBeta、Web hooks）进行实时采集。对互联网数据源支持通过爬虫（Crawler）工具爬取。数据采集后，通过 Apache Kafka 消息中间件进行消息缓存和中转分发，并借助 Logstash 进行预处理和索引，通过 ES 进行分片存储检索，并基于 Spark 的 Map 和 Reduce 实现开发者能力特征提取、开发者交互行为数据提取、相似度、匹配度计算、矩阵分解拟合计算、多特征融合计算等，最终实现多特征融合的协同过滤推荐和推荐列表展现。

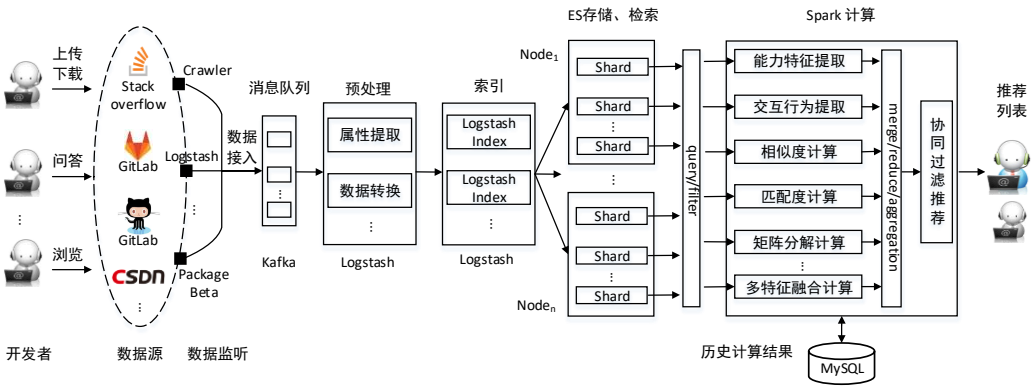


Fig.4 System implementation framework

图 4　系统实现框架

4　实验结果与分析

4.1　数据集及实验设置

为验证本文提出的多特征融合开发者推荐方法的有效性，分别选取互联网技术问答社区 StackOverflow 数据集和东软集团企业内部 GitLab 数据进行了分析，数据集描述如表 4 所示。实验过程采取分组进行的方式，将数据等分为 10 组，采用交叉验证的方式，即对数据集划分为 10 等份，每次选取一组数据作为测试集，其他组作为训练集，最终取平均值。系统软硬件环境配置如表 5 所示，核心参数设置如表 6 所示。

Table 4 Experimental data sets description

表 4　实验数据集描述

数据集	描述
StackOverflow	25.3711 万条可用数据，包括 8.3446 万个提问，26.3532 万个回答，1.6539 万个开发者
GitLab	974.6931 万条记录，其中包括 280 多个 project，130 余位开发者，issue 数 1.6768 万个

Table 5 Experimental environment setting

表 5　实验环境设置

集群	CPU	内存	硬盘	操作系统
Node*4	Intel(R) Core(TM)i5-6500 CPU 3.20GHz	8GB	500G	Windows 7 64 位

Table 6 Experimental parameter setting

表 6　实验参数设置

Parameter	σ	λ	η_0	λ_1	λ_2	λ_3
Value	0.5	0.025	0.00005	0.4	0.3	0.3

4.2　数据特征映射

本节给出开发者能力特征模型与真实数据源之间的特征实例化映射和计算过程描述。为保证特征实例化的合理性，通过参考对东软集团多位技术专家的建议，确定出特定数据源下的开发者能力特征映射方法及相

应权重设置，具体过程描述如下。此外，关于多变量权重的最优化取值问题是下一步工作的研究内容之一，本文仅给出了基于经验值和枚举试验的方式对不同权重进行调整，以试图尽可能的保证权重设置的合理性。特征映射的选取方面，根据历史经验，由于开源社区与企业内部在数据结构和对开发者关注点不同，开源社区往往更加注重开发者的声誉和影响程度，而企业由于人员规模相对较小，往往侧重于开发者工作效率、行为活跃度等特征，因此本文在 StackOverflow 和 GitLab 的开发者特征映射和选取上的侧重点也有所不同。

（1）StackOverflow 特征映射

通过分析 StackOverflow 数据集和实体结构关系，并结合定义 1 给出基于 StackOverflow 的开发者能力特征模型的特征映射实例化。假设能力特征 C={ c_1：“开发者声誉”，c_2：“影响范围”， c_3：“被关注度”}，对应权重 W={ w_1：0.5，w_2：0.2，w_3：0.3}。其中 $rep(u)$，$imp(u)$， $pv(u)$分别映射 StackOverflow 的 User 实体对应的 *reputation*、*impact*、*profile view* 属性，|U|表示开发者集合 U 的元素数。能力特征计算方法为：

$c_1 \leftarrow \frac{rep(u)}{\sum_{i \in U} rep(i)}$， $c_2 \leftarrow \frac{imp(u)}{|U|}$， $c_3 \leftarrow \frac{pv(u)}{|U|}$。

对初始评价矩阵 R^{init} 的映射计算，基于 StackOverflow 的任务-回答者的投票值进行了归一化处理，计算方法为：$r^{init}_{tu} = \frac{vote(t,u)}{\sum_{u \in U} vote(t,u)}$，其中 $vote(t,u)$为开发者 *u* 关于任务 *t* 获得的评分值。

对开发者与任务关联度 $ad(u,t)$的映射计算，设开发者与任务的交互项集合为 A= { a_1：“回答”、a_2：“收藏”、a_3：“评论”、a_4：“浏览”}，假设对应权重α={ α_1：0.4，α_2：0.2，α_3：0.3，α_4：0.1}。其中 $ad(u,t)$的计算方法见定义 3 中的描述。对开发者与任务的匹配度 $md(u,t)$计算过程见定义 2 中的描述。

（2）GitLab 特征映射

同理，给出 GitLab 的数据源特征映射及实例化。能力特征 C={ c_1：“活跃度”，c_2：“任务参与度”， c_3：“贡献度”}，对应权重设为 W={ w_1：0.2，w_2：03，w_3：0.5}。假设 $sigin(u)$、$assign(u)$、$push(u)$表示映射到 GitLab 上 Users 实体的 *sign_in_count*（登录次数）属性、Issues 实体的 *assignee_id*（任务执行者）属性、Events 实体的 *push*（代码提交）属性上的计算方法。其中，$sigin(u)$表示开发者 *u* 的登录次数，$assign(u)$表示开发者 *u* 被指定任务指派者的次数，$push(u)$表示开发者 *u* 代码提交数。计算方法分别为：

$c_1 \leftarrow \frac{sigin(u)}{\sum_{i \in U} sigin(i)}$， $c_2 \leftarrow \frac{assign(u)}{\sum_{i \in U} assign(i)}$， $c_3 \leftarrow \frac{push(u)}{\sum_{i \in U} push(i)}$。对开发者与任务关联度 $rd(u,t)$的映射计算，开发者与任务的交互项集合 A={ a_1：“评论”、a_2：“@”、a_3：“点赞”、a_4：“浏览”}，对应权重α={ α_1：0.4，α_2：0.4，α_3：0.1，α_4：0.1}。计算方法见定义 3 中的描述。对开发者与任务的匹配度 $md(u,t)$，由于本实验项目的 GitLab 软件版本不支持开发者标签属性，所以对 GitLab 开发者在数据预处理阶段进行了标注处理。

4.3 评测方法确立

实验采用离线评测的方式进行，评价指标选取推荐系统中通用的平均准确度(Mean Average Precision, MAP) 和覆盖率（Coverage）评价指标。MAP 评价指标主要用于衡量推荐结果排序的好坏程度，其中，准确度指标（*ap*）的计算方法为公式（12）所示，*len* 表示推荐列表长度，$prediction_i$ 表示前 *i* 个推荐结果的正确率，$(change\ in\ recall)_i$ 为二值项，当第 *i* 个推荐结果正确时（推荐结果的正确性判定，是通过将分组实验的推荐结果与测试集中的真实结果对比得出），值为 1/*len*，否则为 0。最终平均准确度 MAP 是对所有推荐结果的（*ap*）取平均值，关于公式（12）的详细说明见文献[24]。

$$ap@len = \sum_{i=1}^{len} \left(prediction_i \times (change\ in\ recall)_i \right) \quad (12)$$

Coverage 评价指标主要衡量推荐结果的覆盖范围，尽可能让更多的开发者都有机会被推荐，避免只有少

数的开发者被推荐的而出现的“马太效应”。Coverage 评价指标定义为被推荐的开发者数量占开发者总数的比例。其计算过程如公式（13）所示，T 为任务集合，U 为开发者集合，$u(t)$表示推荐给任务 t 的开发者集合。

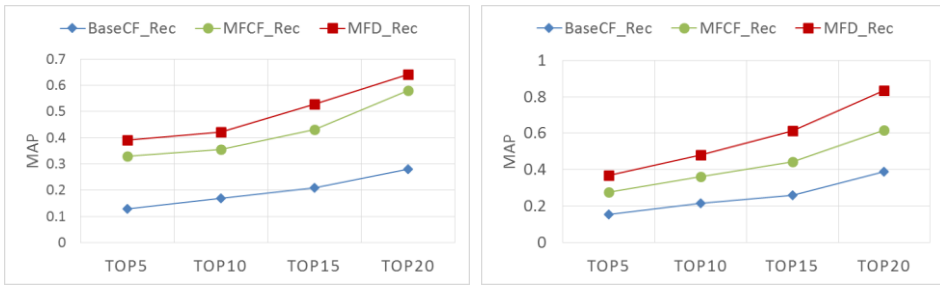
$$coverage = \frac{\left|\bigcup_{t\in T} u(t)\right|}{|U|} \tag{13}$$

4.4 实验结果分析

通过实验分别给出本文所提出的多特征融合协同过滤方法与目前典型的协同过滤推荐方法的对比分析。其中，BaseCF_Rec 是基于原始评价矩阵 R^{init} 实现的协同过滤开发者推荐方法，作为基线对比算法，协同过滤推荐算法原理参见文献[14]。MFCF_Rec 是基于矩阵分解（R^{ad}）的协同过滤推荐算法，MFCF_Rec 未考虑用户能力与行为感知对评价矩阵的增强作用，MFCF_Rec 是目前协同过滤推荐算法中典型的算法实现，矩阵分解算法原理参见文献[4]和[8]中的描述。BaseCF_Rec 与 MFCF_Rec 方法的区别在于，BaseCF_Rec 针对原始评价矩阵进行协同过滤计算，而 MFCF_Rec 方法则是在原始评价矩阵基础上先进行了矩阵分解拟合，在拟合后的矩阵上再进行协同过滤计算，因为 MFCF_Rec 借助矩阵分解拟合原理降低了矩阵稀疏性，所以使得其推荐效果相比 BaseCF_Rec 要好。而本文提出的 MFD_Rec 方法与 BaseCF_Rec 与 MFCF_Rec 的区别在于 MFD_Rec 是基于能力与行为感知多特征融合评价矩阵（R^{mfd}）增强的评价矩阵基础上进行的协同过滤开发者推荐方法。

（1）平均准确度 MAP 分析

如图 5 所示，分别基于 StackOverflow 和 GitLab 数据集进行推荐算法的平均准确率指标 MAP 对比分析。由于 BaseCF_Rec 基于原始评价矩阵 R^{init} 进行的协同推荐算法，原始评价矩阵的稀疏问题导致了平均推荐准确度非常低（StackOverflow 为 0.19，GitLab 为 0.25）。而本文提出的多特征融合增强的推荐方法 MFD_Rec 相比 BaseCF_Rec、MFCF_Rec 算法，具有非常好的平均推荐准确度（StackOverflow 为 0.43，GitLab 为 0.67）。另外，StackOverflow 相比 GitLab 数据源而言，平均准确度相对较低，主要原因是企业内部 GitLab 数据集中的开发者数量较少，但网络抓包数据中存在大量用户访问交互的行为数据，这使得开发者与 issue 的交互行为产生的关联关系非常紧密，开发者与任务的交互行为解决评价矩阵的数据稀疏问题起到重要作用。



(a) StackOverflow (b) GitLab

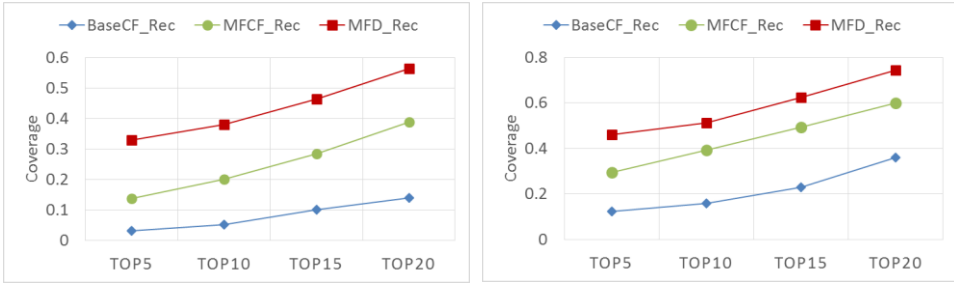
Fig.5 Comparison and analysis of MAP indexes

图 5 MAP 指标对比分析

（2）平均覆盖率 Coverage 分析

如图 6 所示，是对平均覆盖率的对比分析，由于对评价矩阵从多个特征维度的增强，使得评价矩阵中的未知项所占的比例平均减小了 78.43%。对于 StackOverflow 数据集，MFD_Rec 相比 MFCF_Rec 覆盖率平均提升约 1.71 倍，相比 BaseCF_Rec 覆盖率平均提升约 2.36 倍。而对于 GitLab 数据源而言，MFD_Rec 相比 MFCF_Rec 的覆盖率平均提升约 1.31 倍，相比 BaseCF_Rec 覆盖率平均提升约 3.42 倍。由于 StackOverflow 评价矩阵相比 GitLab 评价矩阵更为稀疏，使得对 StackOverflow 运用多特征融合增强后的平均覆盖率提升效果也比较显著。这表明，MFD_Rec 方法与 MFCF_Rec、BaseCF_Rec 方法相比，推荐结果的覆盖率提升也比较明

显。



(a) StackOverflow (b) GitLab

Fig.6 Comparison and analysis of Coverage indexes

图 6 Coverage 指标对比分析

（3）特征权重因子影响分析

为进一步分析不同特征权重对推荐结果的影响，基于 StackOverflow 数据集分别对权重因子λ_1，λ_2，λ_3给出不同的权重组合，通过权重因子递增或递减调整验证对推荐结果的影响。如表 7 所示，给出 TOP10 推荐过程中的结果分析，第 1 行作为基线对比，从序号 2 对应数据可见，当固定λ_1为 0.33， 增加λ_2为 0.60，λ_3调整为 0.07 时，评价指标 MAP 和 Coverage 分别下降 35.34%和 37.18%，这是由于数据集中存在大量能力特征值很高的开发者，但这些开发者又都与特定任务的关联关系和匹配度非常小（遵循复杂网络的无标度特征），因此，如果给予λ_2过高权重，可能导致每次只有少数的能力特征值高的开发者被推荐，即所谓的马太效应。对序号 4，当固定λ_2为 0.33， 增加λ_1为 0.60，λ_3调整为 0.07 时，评价指标 MAP 和 Coverage 分别上升 7.69%和 17.78%，主要是因为增加了用户动态行为关联关系和矩阵分解对协同过滤推荐增强的结果。对序号 7，当固定λ_3为 0.33， 增加λ_1为 0.60，λ_2调整为 0.07 时，评价指标 MAP 和 Coverage 分别上升 12.82%和 26.67%，可见， 给予λ_1、λ_3适当高的权值，能够提升推荐结果的准确率和覆盖率。对于开发者个体能力特征，通常大型 IT 企业更关注开发者个体能力，例如面试时往往会参考开发者在开源社区的贡献和影响力等特征指标，即使被推荐的开发者与项目或者任务的匹配度、关联度不高，但企业仍愿意承担短期的培训成本。

Table 7 The influence of weighting factor on MFD_Rec evaluation index

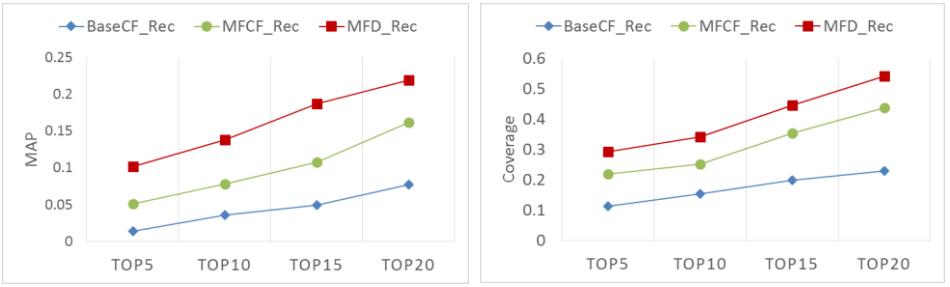
表 7 权重因子对 MFD_Rec 评价指标的影响

序号	权重因子			评价指标	
	λ_1	λ_2	λ_3	MAP	Coverage
1	**0.33**	**0.33**	**0.33**	**0.39**	**0.45**
2	**0.33**	**0.60**	**0.07**	**0.25**	**0.29**
3	0.33	0.07	0.60	0.41	0.47
4	**0.60**	**0.33**	**0.07**	**0.42**	**0.53**
5	0.07	0.33	0.60	0.32	0.36
6	0.07	0.60	0.33	0.23	0.31
7	**0.60**	**0.07**	**0.33**	**0.44**	**0.57**

（4）冷启动案例分析

为证明本文推荐方法对冷启动问题的效率提升，选取 GitLab 上新创建的状态为“open”且未指派负责人的 105 个典型 issue 进行冷启动问题案例分析。由于选取的案例对应初始评价矩阵 R^{init} 的数据非常稀疏，导致现有协同过滤推荐方法的准确率和覆盖率都很低，但通过对开发者行为数据分析发现，新创建的 issue 中，约

32.19%的 issue 都有被不同开发者点赞、浏览、评论、‘@’等行为数据存在，且不同开发者与 issue 关联关系的强弱程度也各不相同。基于本文定义的 GitLab 特征映射计算方法，并根据定义 3 中给出的开发者行为关联关系对开发者进行推荐算法生成，针对选取的样本数据的进行对比试验分析的结果如图 7 所示。从图 7 可以看出，即便在评价矩阵稀疏的情况下， MFD_Rec 方法由于借助了开发者能力和行为感知的特征融合方法相比现有技术方案具备明显优势。冷启动状态下 MFD_Rec 的平均准确率为 0.16，平均覆盖率为 0.45，相比 BaseCF_Rec 的平均推荐准确率提升 2.67 倍，平均覆盖率提升 1.33 倍， 相比 MFCF_Rec 的平均推荐准确率提升 63.13%，平均覆盖率提升 80.96%。从 GitLab 的冷启动案例的分析结果可以看出，MFD_Rec 对冷启动问题的推荐准确度和覆盖率相比现有方法具有明显改善。



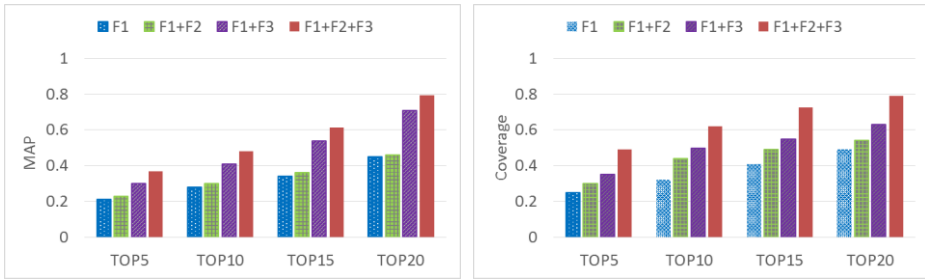
(a) MAP (b) Coverage

Fig.7 Case analysis of cold start problems

图 7 冷启动问题案例分析

（5）不同特征组合分析

为进一步分析单一特征和多特征组合下算法效率，基于 GitLab 数据集对不同特征组合进行对比分析，假设 F1、F2、F3 分别表示公式（10）中的 r^{mfd}_{tu}、$dcm(u)$、$md(u,t)$对应的特征项。如图 8 所示，F1+F2 相比 F1，平均准确度指标非常接近，但平均覆盖率指标却有所提升，这表明单纯的 $dcm(u)$对平均准确度影响较小，但对覆盖率却有很好的提升作用。F1+F3 相比 F1+F2 具有相对好的平均准确度和覆盖率，这表明单纯的开发者与任务匹配度对推荐结果的影响大于开发者能力特征的影响，而 F1+F2+F3 相比 F1、F1+F2 或 F1+F3 来说具有更好的推荐准确度和覆盖率。可见，F1+F2+F3 相比单一特征或单一特征组合的实现方式具有很好的效果。



(a) MAP (b) Coverage

Fig.8 Analysis of different combination of features

图 8 不同特征组合案例分析

5 总结

随着开源共享、群体协作与竞争趋势的日益显现，以兴趣和自组织协作开发模式为基础的开源社区、以自愿精神和知识共享为基础的开发者社区、以金钱激励竞争性任务分发机制为基础的众包软件等群体软件开

发模式成为软件工程领域的新特点。在这一趋势下，开发者能力评价和协作关系推荐开始受到学术界和产业界的重视。如何有效利用软件过程大数据资源对开发者能力及开发者之间的协作关系进行建模、分析和挖掘，从价值密度稀疏、体量巨大的软件过程大数据中挖掘有价值的知识，为互联网社区、群体软件开发和 IT 企业提供个性化、全视角的开发者、开发资源、协作关系等方面的精准推荐服务，是提升群体软件工程开发与协作效率的新挑战。

本文从开发者推荐系统实践过程中存在的开发者能力评价问题、数据稀疏问题、大数据环境下推荐系统构建问题出发，提出基于模糊综合评价的开发者能力评价模型，在此基础上，借助矩阵分解技术对评价矩阵增强优化，提出一种能力和行为感知的多特征融合协同过滤开发者推荐方法，有效解决了开发者推荐过程中存在的数据稀疏性、冷启动等问题。从工程实践角度出发，给出面向大数据环境的推荐技术方案选型、系统实现及优化过程的阐述。最后，对互联网问答社区 StackOverflow 数据集和东软集团企业内部 GitLab 的真实生产环境进行了实验分析，从不同评价指标维度给出了本文方法的有效性证明。

未来工作，研究面向特定应用领域大项目、跨部门协作环境下的开发者协作团队推荐模型，构建协作关系在线实时推荐服务框架。解决目前 IT 企业 DevOps 开发运维一体化支撑普遍面临的多项目、大规模、多版本、跨团队协作开发、持续交付过程中面临的协作效率低下、沟通成本过高的问题。

致谢

本文实验分析、论文撰写和系统实践过程中得到了东软集团基础软件事业部架构师王伟、研发工程师王汉冲等人的支持和帮助，在此一并表示感谢。

References:

[1] Li GL, Wang JN, Zheng YD, Franklin JM. Crowdsourced data management: A survey. ACM Trans. on Knowledge and DataEngineering, 2016, 28(9):2296−2319. [doi: 10.1109/TKDE.2016.2535242]

[2] Robert Armstrong, Dayne Freitag, Thorsten Joachims, et al. Web Watcher: A learning apprentice for the World Wide Web. In: Proceedings of the AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments, 1995. 6-12. [doi: 10.21236/ada640219]

[3] Yanxiang Huang, Bin Cui, Jie Jiang, et al. Real-time Video Recommendation Exploration. ACM SIGMOD Conference, 2016. 35-46. [doi: 10.1145/2882903.2903743]

[4] Yanxiang Huang, Bin Cui, Wenyu Zhang, et al. TencentRec: Real-time Stream Recommendation in Practice. ACM SIGMOD Conference, 2015. 227-238. [doi: 10.1145/2723372.2742785]

[5] Hongzhi Yin, Bin Cui, Ling Chen, et al. A Temporal Context-Aware Model for User Behavior Modeling in Social Media Systems. ACM SIGMOD Conference, 2014. 1543-1554. [doi: 10.1145/2588555.2593685]

[6] Xiangmin Zhou, Lei Chen, Yanchun Zhang, et al. Online Video Recommendation in Sharing Community. ACM SIGMOD Conference, 2015. 1645-1656. [doi: 10.1145/2723372.2749444]

[7] Chen Chen, Hongzhi Yin, Junjie Yao, et al. TeRec: A Temporal Recommender System Over Tweet Stream. VLDB Endowment, 2013. 1254-1257. [doi: 10.14778/2536274.2536289]

[8] Royi Ronen, Elad Yom-Tov, Gal Lavee. Recommendations meet web browsing enhancing collaborative filtering using internet browsing logs. ICDE Conference, 2016. 1230-1238. [doi: 10.1109/icde.2016.7498327]

[9] Cheng Li, Yue Lu, Qiaozhu Mei, et al. Click-through Prediction for Advertising in Twitter Timeline. ACM SIGKDD Conference, 2015. 1959-1968. [doi: 10.1145/2783258.2788582]

[10] Hongzhi Yin, Yizhou Sun, Bin Cui, et al. LCARS: A Location-Content-Aware Recommender System. ACM SIGKDD Conference, 2013. 221-229 .[doi: 10.1145/2487575.2487608]

[11] Yuchen Li, Dongxiang Zhang, Ziquan LAN, et al. Context-Aware Advertisement Recommendation for High-Speed Social News Feeding. ICDE Conference, 2016. 505-516 .[doi: 10.1109/ICDE.2016.7498266]

[12] Won-Seok Hwang, Juan Parc, Sang-Wook Kim, et al. “Told You I Didn’t Like It”: Exploiting Uninteresting Items for Effective Collaborative Filtering. ICDE Conference, 2016. 349-360.[doi: 10.1109/icde.2016.7498253]

[13] Bo Jiang, Jiguang Liang, Ying Sha, et al. Retweeting Behavior Prediction Based on One-Class Collaborative Filtering in Social Networks. ACM SIGIR Conference, 2016. 977-980.[doi: 10.1145/2911451.2914713]

[14] Rong Pan, Yunhong Zhou, Bin Cao, et al. One-Class Collaborative Filtering. In Proc. of IEEE ICDM, 2008.502–511. [doi: 10.1109/icdm.2008.16]

[15] Hans-Jörg Happel, Walid Maalej. Potentials and challenges of recommendation systems for software development. International Workshop on Recommendation Systems for Software Engineering, 2008, (3):11-15. [doi: 10.1145/1454247.1454251]

[16] Martin P. Robillard, Robert J. Walker. An Introduction to Recommendation Systems in Software Engineering. Recommendation Systems in Software Engineering, 2013.1-11. [doi: 10.1007/978-3-642-45135-5_1]

[17] Xin Xia, David Lo, Xinyu Wang, et al. Accurate Developer Recommendation for Bug Resolution. Working Conference on Reverse Engineering (WCRE), 2013.72-81. [doi: 10.1109/wcre.2013.6671282]

[18] J Tang, S Wu, J Sun, et al. Cross-domain Collaboration Recommendation. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2012 .1285-1293. [doi: 10.1145/2339530.2339730]

[19] K Mao, Y Yang, Q Wang, et al. M HarmanDeveloper Recommendation for Crowdsourced Software Development Tasks. Service-oriented System Engineering, 2015.347-356. [doi: 10.1109/sose.2015.46]

[20] Y Tian, PS Kochhar, EP Lim, et al. Predicting Best Answerers for New Questions: An Approach Leveraging Topic Modeling and Collaborative Voting, SocInfo Workshops, 2013.55-68. [doi.org/10.1007/978-3-642-55285-4_5]

[21] Ke Mao, Ye Yang, Qing Wang, et al. Developer Recommendation for Crowdsourced Software Development Tasks. Service-oriented System Engineering, 2015.347-356. [doi: 10.1109/sose.2015.46]

[22] Wendi Niu, Hong Zhang. A preference-based recommendation method with fuzzy comprehensive evaluation.Intelligent Decision Technologies, 2014, 8(3): 179-187. [doi: 10.3233/idt-140187]

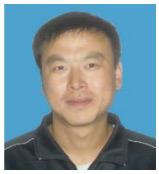
[23] ZHANG Yu-Jie, DU Yu-Lu, MENG Xiang-Wu. Research on Group Recommender Systems and Their Applications. Chinese Journal of Computers, 2016, 39(4):746−764 (in Chinese with English abstract). http://cjc.ict.ac.cn/qwjs/view.asp?id=4638

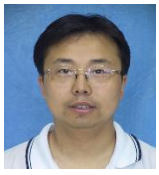
[24] NIU Wen-Jia, LIU Ji-Qiang, SHI Chuan, et al. User’s network behavior portrait –The portrait analysis of user’s network behavior and content recommendation applications in the field of bigdata.Publishing House of Electronics Industry, 2016:154−162(in Chinese)

附中文参考文献:

[23] 张玉洁,杜雨露,孟祥武.组推荐系统及其应用研究.计算机学报,2016,39(4):746−764

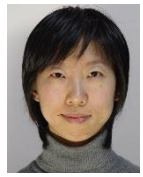
[24] 牛温佳,刘吉强,石川,等.用户网络行为画像—大数据中的用户网络行为画像分析与内容推荐应用.电子工业出版社,2016:154−162





谢新强（1982-），男，博士研究生、研究员、高级软件工程师，CCF 会员，主要研究领域为推荐系统、大数据分析、软件复用、可信计算等。

王斌（1972-），男，副教授，主要研究领域为数据处理及优化、分布式数据管理等。



杨晓春（1973-），女，教授，博士生导师，CCF 高级会员，主要研究领域为数据库技术和理论，数据质量分析等。