基于多关系知识增强的开发者推荐算法

杜军威,王昭哲,于 旭,胡 强,江 峰,巩敦卫
(青岛科技大学信息科学技术学院,山东青岛 266061)

摘 要: 近年来,随着众包平台的不断发展,信息过载问题日趋严重,任务难以及时找到可靠的开发者完成,为任务推荐合适的开发者变得至关重要. 传统推荐方法存在两大挑战:一是任务和开发者的文本特征高度简练,传统推荐方法聚焦于表面文本信息,未发现其中包含的大量知识实体;二是任务具有一次性,导致显式交互数据极其稀疏. 为了解决上述挑战,本文提出一种基于多关系知识增强的开发者推荐算法. 对于一个任务和开发者,首先将他们包含的文本内容中的每个单词与知识图谱中的相关实体关联起来,用于丰富任务和开发者的信息表示. 除直接相关联的实体外,还使用每个实体的上下文实体集合来提供更多的信息. 然后,对于开发者本文使用多关系邻域聚合的方式增强其特征表示,并使用注意力模块区分开发者对任务的关注度. 最终获得的用户和开发者的嵌入输入到深度神经网络中进行预测. 在真实的Topcoder数据集上进行广泛的实验,结果表明,本文方法在正确率和序位倒数均值上相比于最佳对比方法平均提高11.7%和17.5%.

关键词: 开发者推荐;软件众包开发;多关系;知识图谱;图神经网络

基金项目: 国家自然科学基金(No.62172249,No.61973180);山东省自然科学基金(No.ZR2021MF092,No.ZR2022MF326);中央高校基本科研业务费(No.93K172022K01)

中图分类号: TP311.5 **文献标识码:** A **文章编号:** 0372-2112(2023)11-3111-09

电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20230271

A Developer Recommendation Algorithm Based on Multi-Relationship Knowledge Enhancement

DU Jun-wei, WANG Zhao-zhe, YU Xu, HU Qiang, JIANG Feng, GONG Dun-wei
(*School of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, Shandong* 266061, *China*)

Abstract: In recent years, crowdsourcing software development has gradually become an emerging software development model. However, with the continuous development of crowdsourcing platforms, the problem of information overload has become increasingly serious. It has become crucial to recommend suitable developers for tasks that are difficult to find reliable developers to complete on time. Traditional recommendation methods face two major challenges: First, the text features of tasks and developers are highly concise. Traditional recommendation methods focus on surface text information and fail to discover the large amount of knowledge entities contained therein. Second, tasks are one-time and this results in extremely sparse explicit interaction data. To solve these challenges, we propose a developer recommendation algorithm based on multi-relational knowledge enhancement. We identify entities and contextual entities from text features and link the relationship between tasks and developers from a knowledge perspective, uncovering the deep preferences of developers. In addition, we treat developers' participation in registering, submitting, and winning tasks as different preferences. We assign different weights to the relationships between tasks and developers and employ an attention mechanism to refine the importance of these different relationships. Finally, we enhance developers' feature representation using multi-relational neighborhood aggregation. We conduct extensive experiments on the real-world Topcoder dataset, and the results show that our method outperforms the best baseline method by an average of 11.7% in accuracy and 17.5% in mean reciprocal rank.

Key words: developer recommendation; crowdsourced software development; multi-relationship; knowledge graph; graph neural network

Foundation Item(s): National Natural Science Foundation of China (No.62172249, No.61973180); Natural Science

收稿日期:2023-03-28;修回日期:2023-09-22;责任编辑:覃怀银

Foundation of Shandong Province (No.ZR2021MF092, No.ZR2022MF326); Basic Research Fund for Central Universities (No.93K172022K01)

1 引言

近年来,群体协作开发软件逐渐成为一种新兴的软件开发模式[1]. 这不仅提高了软件开发质量,降低了开发成本,还极大地满足了人们对软件的需求. 众包软件平台为群体协作开发提供了便利渠道,通过将软件开发任务外包给众包开发人员来获得高品质的软件产品[2]. 随着众包平台的不断发展,平台上积累的开发者和发布的任务数目变得十分庞大,信息过载问题日趋严重,任务难以及时找到可靠的开发者完成. 同时,开发者的偏好也影响任务的完成,从事不理想的任务会产生消极的后果[3]. 因此,为任务推荐合适的开发者变得至关重要,成为近期的研究热点.

为了推荐合适的开发者,研究人员做出了大量努力. 例如,Mao等人[4]提出一种基于内容的分类方法推荐开发者. Shao等人[5]通过结合神经网络和潜在语义索引模型推荐开发者. Zhu等人[6]根据描述信息构建排序学习模型实现开发者的有效推荐. Zhang等人[7]提出一种基于元学习的策略模型推荐可能获胜的开发者. 上述方法简单地使用描述信息表示开发者和任务的特征,未挖掘深层次关联关系,模型的推荐性能不够理想. 此外,上述方法舍弃了开发者参与和提交任务的信息对于开发者偏好的影响,未能全面捕捉开发者的兴趣偏好,在开发者获胜次数较少时,推荐性能不佳.

为了发现任务和开发者之间的潜在联系,扩展开发者的兴趣偏好,我们在开发者推荐中引入知识图谱[8]来提取任务和开发者之间的深层联系. 通过建立知识图谱,我们可以从实体之间的关系中发现深层联系,从而推荐对任务更感兴趣的开发者. 同时,知识图谱可以通过链接不同实体之间的关系,进一步扩展开发者的兴趣偏好.

此外,为了解决任务和开发者之间交互数据极度稀疏的问题,除了考虑开发者在平台上的获胜信息,我们还将开发者在任务注册和提交阶段的信息视为开发者与任务的交互. 本文设计一种多关系邻域聚合的方式,赋予注册、提交、获胜三种关系不同的权重来全面感知开发者偏好和能力. 将开发者的表示与任务的表示联系起来,通过图卷积神经网络(Graph Convolutional Neural Network,GCN)聚合局部邻域的特征.

因此,本文提出了一种多关系知识增强的开发者推荐算法(Multi-relationship Knowledge-enhanced Developer Recommendation algorithm,MKDR). 它将一个新任务和一个开发者(包含新开发者)作为输入,输出任务对开发者的评价. 为了验证所提方法的有效性,我们在真实的数据集上进行了实验,实验结果表明MKDR在开发者推荐上明显优于其他对比方法.

2 相关工作

2.1 开发者推荐

为了更准确地推荐开发者,一些研究者根据开发者的获胜数据、开发者与任务的特征采取不同的策略进行推荐[4,6,9~11]. 此外,考虑到开发者与任务的不同关系,Zhang等人[7]提出一种基于元学习的策略模型,预测开发者注册、提交和获胜的可能性,逐步过滤不可能获胜的开发者. Yang等人[12]利用关键因素构建动态的众包开发者决策支持模型,预测开发者注册、提交和获胜的可能性. Zhang等人[13]将开发者分为三类,采用多模态学习对三类开发者进行评分预测,然后使用集成学习对评分加权相加来推荐最终的开发者. 但以上方法均依赖于任务和开发者的表面特征,且未考虑开发者注册和提交的任务所包含的偏好,推荐性能不够理想.

2.2 知识图谱

为了提高推荐的准确性,缓解数据稀疏性问题,模型可以利用一些附加信息. 最近,知识图谱表示被证明建模这类数据来增强推荐性能[14,15]. 知识图谱是一个异构网络,包含多种类型的节点和关系,其中节点对应于实体(任务或开发者),边对应于关系. 推荐系统中使用知识图谱信息主要有三种方法:基于嵌入的方法[16,17],基于路径的方法[18,19]和统一的方法[15,20]. MKDR中利用基于嵌入方法中的翻译距离模型[21~24]学习每个实体和关系的低维向量. 目前,知识图谱已经广泛应用在多个场景,包括推荐系统[17]、问答系统[25]、关系检测[26]. 本文将知识图谱嵌入应用在开发者推荐中的工作.

3 基于多关系知识增强的开发者推荐

本文提出一种多关系知识增强的开发者推荐算法,即MKDR. 如图1所示,该算法包含特征处理、知识增强、开发者多关系聚合和结果预测四个部分. 我们首先对任务和开发者的文本特征和非文本特征进行处理,然后提取文本特征中的知识实体和上下文实体. 接下来构建开发者与任务二部图. 在二部图中,开发者与不同关系的任务相关联,聚合这些任务的特征增强开发者的表示. 将任务的嵌入表示和开发者的嵌入表示连接后输入深度神经网络(Deep Neural Network,DNN)中,最终输出预测的结果. 下面将对算法进行详细介绍.

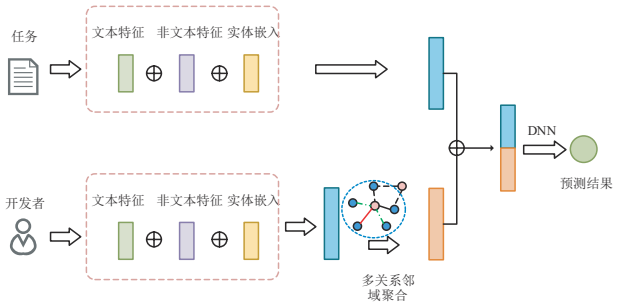

图1　MKDR模型结构

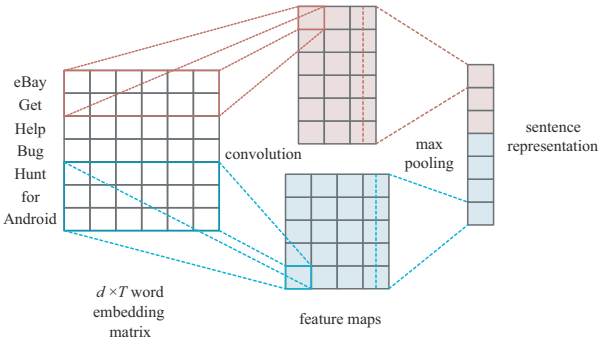

图2　CNN文本特征提取

3.1　问题描述

本文的场景是众包软件开发平台上开发者的推荐. 以Topcoder作为众包软件开发平台为例,开发的完整过程包括任务发布,开发者注册任务,提交作品,最后评审作品选出获胜的开发者进行奖励.

在众包软件开发平台上有 n 个开发者 $U=\{u_1,u_2,\cdots,u_n\}$ 和 m 个任务 $V=\{v_1,v_2,\cdots,v_m\}$. 对于一个开发者 $u\in U$, $V_r=\{v_1,v_2,\cdots,v_{t_1}\}$, $V_s=\{v_1,v_2,\cdots,v_{t_2}\}$ 和 $V_w=\{v_1,v_2,\cdots,v_{t_3}\}$ 分别表示开发者注册,提交和获胜的历史任务,其中 $\tau_1\geqslant\tau_2\geqslant\tau_3$. 通过聚合历史任务来充分表示开发者特征,最终在给定一个新任务 v 的情况下,从开发者集合 U 中推荐合适的开发者完成任务.

3.2　特征处理

3.2.1　文本特征处理

对于文本特征我们使用基于CNN的句子表示学习模型进行处理,与词袋模型相比[27],基于CNN的句子表示学习模型[28,29]特征提取能力更强. 如图2所示,输入长度为 T 的句子,首先词嵌入层将句子中的每个单词映射为 d 维向量表示 $\left[\boldsymbol{w}_1\ \boldsymbol{w}_2\ \cdots\ \boldsymbol{w}_T\right]\in\mathbb{R}^{d\times T}$,这些词的嵌入表示通过预训练[30]得到,且在整个训练过程中保持不变. 接下来卷积层使用滤波器 $\boldsymbol{k}\in\mathbb{R}^{d\times t}$ 对词嵌入矩阵进行卷积运算,t 是滤波器的窗口大小,通常为2~5. 卷积层第 j 个神经元的产生的特征为

$$z_j=f(\boldsymbol{w}_{1:T}*\boldsymbol{k}_j+\boldsymbol{b}_j)\tag{1}$$

其中,f 是非线性函数,$*$ 是卷积操作,b_j 是偏置. $\left\{z_j^1,z_j^2,\cdots,z_j^{(T-t+1)}\right\}$ 是第 j 个神经元在嵌入文本上的滑动窗口所产生的特征. 对这些特征进行最大池化运算可以识别最重要的特征,该神经元最终的特征为

$$\tilde{z}_j=\max\{z_j^1,z_j^2,\cdots,z_j^{(T-t+1)}\}\tag{2}$$

卷积层的最终输出是其 m 个神经元输出的连接:

$$z=(\tilde{z}_1\ \tilde{z}_2,\cdots,\tilde{z}_m)\tag{3}$$

3.2.2　非文本特征处理

任务和开发者的特征中有类别型特征和数值型特征. 类别特征包括任务所需技术、编程语言、开发者类型、开发者位置和开发者技能. 数值特征包括任务发布日期、奖励、开发者注册日期等. 对于类别特征,本文使用one-hot或者multi-hot编码这些特征[31]:

$$\underbrace{(1,0,1,\cdots,1)}_{\text{techniques or language}}\quad\underbrace{(0,1,\cdots,0)}_{\text{developer type or location}}\quad\underbrace{(0,1,1,\cdots,0)}_{\text{developer skill}}\tag{4}$$

对于数值特征,我们采用Z-score归一化方法处理. 最终将这些文本特征和非文本特征拼接在一起.

3.3　知识图谱增强表示

任务的标题和开发者的自我介绍等文本特征中包含着大量的知识实体,实体间通过链接可以发现开发者的潜在兴趣,从而增强模型效果. 本文首先基于外部知识图谱DBpedia[8]进行实体链接,然后基于识别的实体构建子图,并获得实体之间的关系链接. 此外,我们可以利用知识图谱上的信息传播和聚合来提升实体语义表示. 因此,我们将已识别实体的上下文实体也加入到子图中. 基于扩展后的子图我们采用知识图嵌入方法[24]用于实体学习,获得的实体嵌入矩阵表示为 $\boldsymbol{E}=[\boldsymbol{e}_1\ \boldsymbol{e}_2\ \cdots\ \boldsymbol{e}_n]$,$n$ 是文本中提到的实体数量. 进一步,本文使用已识别实体的上下文实体来丰富实体嵌入表示. 实体 e 的上下文实体定义为

$$\text{context}(e)=\{e_i|(e,r,e_i)\in G\ \text{or}\ (e_i,r,e)\in G\}\tag{5}$$

其中,r 表示关系,G 表示知识图谱. 给定实体 e 的上下文,上下文实体嵌入表示为

$$\bar{\boldsymbol{e}}=\frac{1}{|\text{context}|}\sum_{e_i\in\text{context}(e)}\boldsymbol{e}_i\tag{6}$$

将实体 e 及其上下文与对应的文本特征进行比较,应用softmax函数为每个实体及其上下文分配不同的重要性:

$$p_i=\text{softmax}(z^{\mathrm{T}}\cdot\tilde{\boldsymbol{e}}_i)=\frac{\exp(z^{\mathrm{T}}\cdot\tilde{\boldsymbol{e}}_i)}{\sum_{\boldsymbol{e}_i\in\boldsymbol{E}}\exp(z^{\mathrm{T}}\cdot\tilde{\boldsymbol{e}}_i)}\tag{7}$$

其中,$\tilde{\boldsymbol{e}}_i=\boldsymbol{e}_i+\bar{\boldsymbol{e}}_i$. 最终,开发者文本实体嵌入的表示为每一个实体嵌入的加权和:

$$\boldsymbol{e}=\sum_{\boldsymbol{e}_i\in\boldsymbol{E}}p_i\tilde{\boldsymbol{e}}_i\tag{8}$$

3.4 多关系邻域聚合

为了挖掘开发者深层次的连接信息，如图3所示，本文构建了任务-开发者多关系二部图，将开发者的表示与任务的表示联系起来. Michael Schlichtkrull 等人[32]证明GCN通过聚合局部邻域的特征可以有效增强模型的性能.

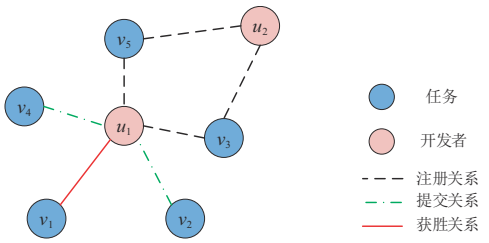

图3 任务-开发者多关系图

开发者和任务之间有注册(register)，提交(submit)和获胜(win)三种关系，每种关系对于开发者表示的影响有所不同. 通过聚合邻域信息后可以得到开发者的特征，如下所示：

$$\boldsymbol{h}_u^{(l+1)}=\sigma(\boldsymbol{W}_1(\boldsymbol{h}_u^{(l)}+\boldsymbol{h}_{I_u}^{(l)})) \tag{9}$$

其中，$\boldsymbol{h}_u^{(l+1)}$，$\boldsymbol{h}^{(l)}$分别是开发者u在l层和l+1层的表示，$\boldsymbol{h}_{I_u}^{(l)}$是开发者$u$的$l$层邻域聚合表示. I_u是开发者u相关联邻居的任务集合，$\boldsymbol{W}_1$是权重矩阵，σ=LeakyReLU(·).

开发者来自与其相交互的邻域特征为

$$\begin{aligned}\boldsymbol{h}_{I_u}^{(l)}=&\boldsymbol{W}_r^{(l)}\sum_{v\in I_{u,r}^v}c_r(u,v)\boldsymbol{h}_v^{(l)}+\boldsymbol{W}_s^{(l)}\sum_{v\in I_{u,s}^v}c_s(u,v)\boldsymbol{h}_v^{(l)}\\&+\boldsymbol{W}_w^{(l)}\sum_{v\in I_{u,w}^v}c_w(u,v)\boldsymbol{h}_v^{(l)}\end{aligned} \tag{10}$$

$$c(u,v)=\mathrm{softmax}(\boldsymbol{h}_u\cdot\boldsymbol{h}_v)=\frac{\exp(\boldsymbol{h}_u\cdot\boldsymbol{h}_v)}{\sum_{v\in I_u^v}\exp(\boldsymbol{h}_u\cdot\boldsymbol{h}_v)} \tag{11}$$

其中，$\boldsymbol{W}_r^{(l)}$，$\boldsymbol{W}_s^{(l)}$和$\boldsymbol{W}_w^{(l)}$分别是衡量注册、提交和获胜关系重要性的权重矩阵. $\boldsymbol{h}_v^{(l)}$是任务v在l层的表示. $I_{u,r}^v$，$I_{u,s}^v$和$I_{u,w}^v$分别是注册、提交和获胜关系下的开发者u关联到的任务集合. $c(u,v)$是任务-开发者之间的归一化分数，可以学习任务v对于开发者u的重要程度，有利于聚合更重要的信息. 类似地，我们可以得到任务表示$\boldsymbol{h}_v^{(l)}$用于中间层的信息传递. 我们设置$\boldsymbol{h}_u^{(0)}=\boldsymbol{h}_u$，$\boldsymbol{h}_v^{(0)}=\boldsymbol{h}_v$用于初始信息传递.

3.5 预测

经过多关系邻域聚合后，我们获得开发者的特征表示$\boldsymbol{h}_u$. 由于需要预测的任务都是新发布的，我们直接对新任务进行处理，获得任务的特征表示$\boldsymbol{h}_v$. 将任务与开发者的特征拼接起来：

$$\boldsymbol{h}=\boldsymbol{h}_u\oplus\boldsymbol{h}_v \tag{12}$$

其中，$\oplus$表示拼接操作. 将特征$\boldsymbol{h}$作为DNN的输入，最终输出预测结果：

$$\hat{y}=\mathrm{DNN}_\Theta(\boldsymbol{h}) \tag{13}$$

其中，Θ表示DNN中的参数.

3.6 模型学习

为了学习MKDR的参数，本文使用推荐系统中广泛使用的逐点对数损失作为目标函数：

$$\mathrm{Loss}=-\sum_{(u,v)\in Y^+\cup Y^-}y\log\hat{y}+(1-y)\log(1-\hat{y}) \tag{14}$$

其中，y表示真实标签1或0(1表示开发者u获胜了任务v，0则表示没有获胜)，$\hat{y}$表示预测的值. Y^+和Y^-分别是正负样本集合.

目标函数的优化可以通过随机梯度下降来完成. 在开发者推荐中，获胜的开发者往往只有一两个人，而参与任务的开发者多达数十人，这导致负样本过大. 因此，本文采取过采样的方法来平衡正负样本，从正样本集中反复随机抽取样本来增加数据集中正样本的数量. 通过过采样，将正样本数量增加到与负样本数量相同，此时模型训练会过于强调正样本，正样本的误差会被放大，模型容易在正样本上出现过拟合. 为了达到更好的算法学习效果，本文设置正负样本比例为1∶5.

MKDR算法的训练过程如算法1所示.

算法1 基于多关系知识增强的开发者推荐算法

输入: 任务的信息v_info, 开发者的信息u_info, 知识图谱G, 开发者与任务的交互信息

输出: 参数Θ, $\boldsymbol{W}$

初始化模型参数
repeat
  for $(u,v)\in Y^+\cup Y^-$ do
    $\boldsymbol{h}_{v\_\mathrm{text}},\boldsymbol{h}_{u\_\mathrm{text}}=\mathrm{CNN\_processor}(v\_\mathrm{info},u\_\mathrm{info})$
    //提取任务和开发者的文本特征
    $\boldsymbol{h}_{v\_\mathrm{notex}},\boldsymbol{h}_{u\_\mathrm{notex}}=\mathrm{Non\text{-}text\_feature\_processing}(v\_\mathrm{info},u\_\mathrm{info})$
    //提取任务和开发者的非文本特征
    $\boldsymbol{e}_v,\boldsymbol{e}_u=\mathrm{Knowledge\_Graph\_Embedding}(v\_\mathrm{info},u\_\mathrm{info})$
    //获得知识图谱嵌入表示
    $\boldsymbol{h}_v=(\boldsymbol{h}_{v\_\mathrm{text}},\boldsymbol{h}_{v\_\mathrm{notex}},\boldsymbol{e}_v)$, $\boldsymbol{h}_u=(\boldsymbol{h}_{u\_\mathrm{text}},\boldsymbol{h}_{u\_\mathrm{notex}},\boldsymbol{e}_u)$
    for l=1: L do
      $\boldsymbol{h}_u^{(l)}=\sigma(\boldsymbol{W}(\boldsymbol{h}_u^{(l-1)}+\boldsymbol{h}_{I_u}^{(l-1)}))$
      $\boldsymbol{h}_{I_u}^{(l)}=\boldsymbol{W}\sum c\boldsymbol{h}_v^{(l-1)}$
    end for
    $\boldsymbol{h}=(\boldsymbol{h}_u^L,\boldsymbol{h}_v^L)$
    $\hat{y}=\mathrm{DNN}(\boldsymbol{h})$
    计算损失函数,更新参数Θ, $\boldsymbol{W}$
  end for
until Convergence
return Θ, $\boldsymbol{W}$

4　实验结果与分析

为了验证所提方法MKDR的有效性,本文使用真实平台Topcoder上的数据进行实验,并给出了相应的实验结果和分析. 我们主要研究以下问题:

RQ1:所提的开发者推荐方法是否优于其他对比方法?

RQ2:知识实体嵌入是否可以增强特征表示?

RQ3:任务与开发者多种关系的信息聚合是否提升了模型性能?

4.1　数据集介绍

Topcoder平台采用基于竞争的任务完成模式,每个开发者通过注册参与完成任务(Topcoder中称为挑战). 本实验中,我们调取相关的竞赛任务数据,包括缺陷搜寻、代码编写、模块测试等,共计32 846个任务. 获取数据后,我们首先去掉不完整的数据,然后对不同类型的任务进行分类来获得不同的数据集. 为了减少数据稀疏性的影响,我们选择四个较大的数据集进行实验,包括12 034个任务,26 662个注册者、7 106个提交者和3 723个获胜者. 此外,本文使用Microsoft Satori为每个数据集构建知识图谱,提取数据集中出现的实体以及上下文实体,详细信息如表1所示.

表1　Topcoder数据集详细信息

数据集	任务	注册开发者	提交开发者	获胜开发者	实体
Assembly	2 659	3 454	782	335	29 830
Bug hunt	1 532	2 431	615	324	20 573
Code	3 145	14 073	4 360	2 301	76 382
First2Finish	4 698	6 704	1 349	763	52 446

4.2　对比方法

为了评估MKDR的性能,本文将与以下七种先进方法进行对比:

(1) LibFM[33]:该方法是经典的基于特征的因子分解模型. 在LibFM中,开发者和任务的特征由两部分组成:属性特征和实体嵌入. 本文将开发者和任务的特征连接起来作为LibFM的输入.

(2) CrowdRex[4]:该方法将开发者的历史任务和新任务作为模型的输入,使用决策树作为分类器来获得最佳推荐结果. 本文将开发者获胜的历史任务和新任务作为CrowdRex的输入.

(3) DCW-DS[12]:该方法构建了一个动态的决策支持模型,使用随机森林作为分类器,将问题建模为单标签3值分类问题. 本文将任务和开发者的特征作为DCW-DS的输入.

(4) CBC-CN[10]:该方法将相似任务聚类,采用朴素贝叶斯分类器,构建竞争网络细化初始结果. 本文对任务聚类后与开发者一起输入到分类模型中.

(5) PolicyModel[7]:该方法是一种基于元学习的策略模型,使用元学习范式来自动学习机器学习算法和阈值参数. 我们采取与原文一致的参数设置,将任务和开发者特征连接后输入到分类器中.

(6) SusRec[13]:该方法将开发者推荐过程分为两个阶段. 在预处理阶段将候选开发者分为三类;在评分阶段采用多模态学习对三种类型的开发者进行评分,然后使用集成学习对评分进行加权来推荐最终的开发者. 为了与本文方法对比,我们将参与注册、提交和获胜的开发者作为三种不同类型的开发者.

(7) DHRec[11]:该方法结合了任务与开发者的显式特征和隐式特征,构建了有效融合因子分解机和矩阵分解的模型. 为了实现最佳的对比效果,本文按照原始文献中给定的参数范围对参数进行调优.

4.3　评价指标

为了评估MKDR,我们采用正确率(accuracy)和序位倒数均值(Mean Reciprocal Rank,MRR)作为模型的评价指标. 正确率公式如下:

$$\text{accuracy@k}=\frac{\sum_{v\in V}\text{correct}(v,k)}{|V|} \tag{15}$$

其中,V表示任务集合. 如果给定任务v,k个开发者中有一个以上的开发者获胜,则$\text{correct}(v,k)=1$;否则$\text{correct}(v,k)=0$. 序位倒数均值反映了我们推荐的开发者在列表中位置的重要性,序位倒数均值越大,模型的性能越好. 序位倒数均值公式如下:

$$\text{MRR}=\frac{1}{N}\sum_{n=1}^{N}\frac{1}{\text{rank}_n} \tag{16}$$

其中,rank_n表示推荐列表中第一个命中的开发者的排列位置. 以推荐5个开发者$[v_1\ v_2\ v_3\ v_4\ v_5]$为例,真实获胜开发者的位置为3,则$\frac{1}{\text{rank}_n}$计算为$\frac{1}{3}$.

4.4　实验设置

由于需要为新任务推荐开发者,本文根据任务的发布时间对任务进行排序,将新发布的任务用于测试. 将每个实验数据集的80%的数据用于训练,20%用于测试.

在实验环节,实验配置为i7-10710 U处理器,NVIDIA® Geforce® MX350显卡,1.1 GHz主频、16 GB内存,64位Windows 10操作系统. 实验环境所需环境为Python 3.9、PyTorch 1.10.1、Numpy 1.20.3. 单个文本类型的特征的词向量维度为50,卷积核数量设置为20个,知识实体嵌入表示维度为50. 本文采用随机梯度下降训练优化模型,学习率设置为0.001. 使用PyTorch实现一个三层的神经网络,包含一个238节点的输入层,一个64节点的隐含层和一个输出层. 隐含层激活函数

是ReLu,输出层使用Sigmoid激活.

4.5 实验结果和讨论

4.5.1 RQ1:所提的开发者推荐方法是否优于其他对比方法?

为了回答RQ1,我们在4个数据集上将所提方法与对比方法进行比较,计算正确率和序位倒数均值.表2展示了所提方法与对比方法在正确率上的比较结果.

对表2的实验结果进行分析,与LibFM、CrowdRex、DCW-DS、CBC-CN、PolicyModel、SusRec和DHRec相比,我们发现MKDR方法:(1)在accuracy@1上分别提高74.8%、85.8%、70.3%、59.0%、13.7%、16.9%和7.0%;(2)在accuracy@5上分别提高138.0%、167.4%、144.8%、112.7%、31.5%、42.3%和15.3%;(3)在accuracy@10上分别提高164.4%、212.4%、176.2%、139.0%、28.1%、26.4%和12.7%.由表2可知,本文提出的多关系知识增强算法在评价指标accuracy@1、accuracy@5和accuracy@10上都有明显提高.

表2 MKDR方法和对比方法的正确率结果

数据集	评价指标	LibFM	CrowdRex	DCW-DS	CBC-CN	PolicyModel	SusRec	DHRec	**MKDR**
Assembly	acc@1	0.147 6	0.130 1	0.152 5	0.163 8	0.201 3	0.209 4	0.225 3	0.236 5
	acc@5	0.209 3	0.157 9	0.191 8	0.229 4	0.352 2	0.326 0	0.379 5	0.438 4
	acc@10	0.282 2	0.203 8	0.262 4	0.322 7	0.525 6	0.515 4	0.554 2	0.607 3
Bug hunt	acc@1	0.130 5	0.123 2	0.138 0	0.143 6	0.286 7	0.257 1	0.292 8	0.321 1
	acc@5	0.152 4	0.143 9	0.160 8	0.171 3	0.367 5	0.358 8	0.382 0	0.447 6
	acc@10	0.212 1	0.185 7	0.195 3	0.222 8	0.512 0	0.543 6	0.578 3	0.661 4
Code	acc@1	0.136 3	0.142 9	0.144 1	0.160 5	0.195 4	0.183 6	0.210 7	0.225 7
	acc@5	0.171 4	0.168 5	0.172 0	0.209 8	0.263 3	0.235 7	0.331 2	0.374 8
	acc@10	0.216 8	0.204 0	0.224 9	0.268 4	0.417 6	0.433 5	0.475 7	0.557 2
First2Finish	acc@1	0.152 8	0.136 4	0.142 6	0.153 1	0.177 5	0.182 4	0.183 3	0.195 0
	acc@5	0.184 2	0.164 1	0.166 8	0.193 0	0.308 7	0.280 9	0.367 4	0.424 1
	acc@10	0.227 9	0.193 5	0.217 2	0.236 6	0.463 8	0.452 9	0.568 1	0.623 5

序位倒数均值的比较结果如表3所示.MKDR方法与LibFM、CrowdRex、DCW-DS、CBC-CN和PolicyModel相比,MRR值分别提高311.4%、407.9%、212.6%、173.6%、36.0%、29.5%和17.5%.

表3 MKDR方法和对比方法的MRR结果

模型	Assembly	Bug hunt	Code	First2Finish
LibFM	0.131 5	0.105 8	0.872 8	0.114 7
CrowdRex	0.082 3	0.112 4	0.071 1	0.089 2
DCW-DS	0.183 0	0.126 8	0.146 4	0.120 5
CBC-CN	0.181 6	0.153 3	0.159 8	0.147 6
PolicyModel	0.283 7	0.347 5	0.337 2	0.305 8
SusRec	0.268 9	0.363 1	0.372 6	0.332 7
DHRec	0.294 0	0.371 4	0.422 3	0.381 4
MKDR	0.334 9	0.393 6	0.521 2	0.483 0

4.5.2 RQ2:知识实体嵌入是否可以增强特征表示?

为了回答RQ2,本文在四个数据集(D)上进行消融实验,设置两个对比方法,如图4所示.本文测试没有实体嵌入表示的MKDR模型(表示为w/o E)和无上下文实体嵌入表示的MKDR模型(表示为w/o context E).图4结果表明,增加实体嵌入表示可以提高模型性能,增强特征的表达能力.此外,上下文实体的加入,有利于丰富实体嵌入表示,可以进一步提高模型性能.

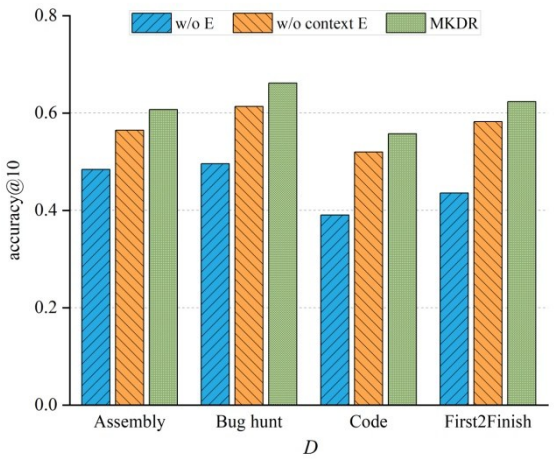

图4 MKDR与两种对比方法在四个数据集上的推荐性能

4.5.3 RQ3:任务与开发者多种关系的信息聚合是否提升了模型性能?

为了回答RQ3,本文对关键参数(即多关系信息聚合层数L)进行研究,在Assembly、Bug hunt、Code和First2Finish数据集上的实验结果如图5所示.图5结果说明,多关系信息聚合是有益的,且当聚合层数为2时模型取得最佳效果.然而,当聚合层数过多会导致信息噪声的增加,影响模型性能.

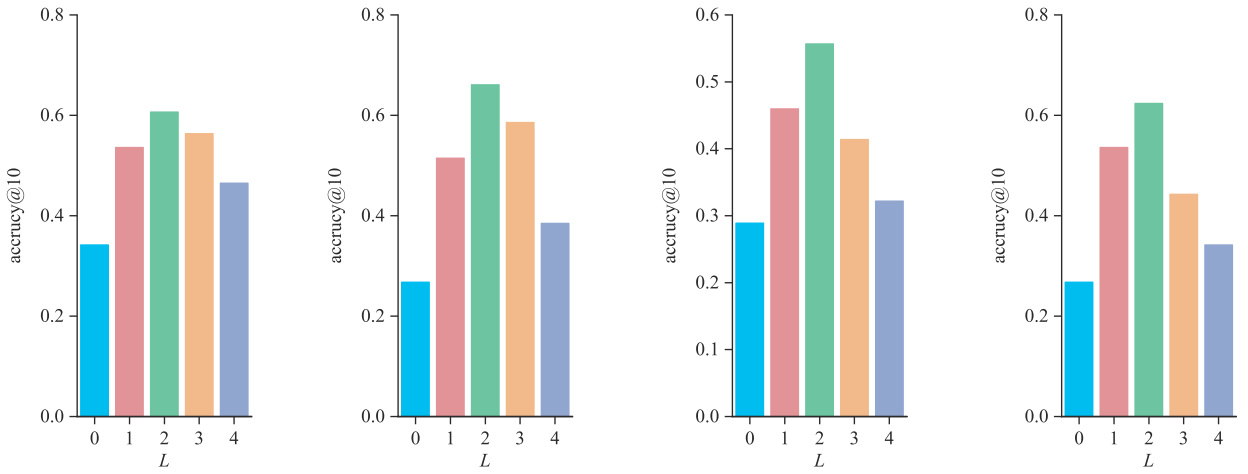

图5　MKDR中不同关键参数的推荐效果

5　总结

在本文中,我们提出一种基于多关系知识增强的开发者推荐算法MKDR. MKDR首次将知识图谱引入开发者推荐中,通过实体链接有效地发现开发者和任务之间的潜在联系. 为了解决开发者与任务的获胜数据稀疏性问题,本文加入开发者与任务的注册和提交关系,在开发者与任务图中聚合高阶邻域信息来增强开发者的表示. 我们对Topcoder平台上的数据集进行了广泛的实验,结果表明,MKDR在正确率和MRR上有着显著的优势. 此外,模型的知识增强模块和多关系聚合模块对模型性能提升提供强大的支撑. 未来,我们考虑从任务角度出发,为开发者推荐合适的任务.

参考文献

[1] ZANATTA A L, MACHADO L, STEINMACHER I. Competence, collaboration, and time management: Barriers and recommendations for crowdworkers[C]//Proceedings of the 5th International Workshop on Crowd Sourcing in Software Engineering. New York: ACM, 2018: 9-16.

[2] 谢新强, 杨晓春, 王斌, 等. 一种多特征融合的软件开发者推荐[J]. 软件学报, 2018, 29(8): 2306-2321.
XIE X Q, YANG X C, WANG B, et al. Multi-feature fused software developer recommendation[J]. Journal of Software, 2018, 29(8): 2306-2321. (in Chinese)

[3] ZAINAB M, RASHINA H, KELLY B, et al. Like, dislike, or just do it? How developers approach software development tasks[J/OL]. Information and Software Technology, 2022, 150. DOI: 10.1016/j.inf-sof.2022.106963.

[4] MAO K, YANG Y, WANG Q, et al. Developer recommendation for crowdsourced software development tasks[C]// 2015 IEEE Symposium on Service-Oriented System Engineering. Piscataway: IEEE, 2015: 347-356.

[5] SHAO W, WANG X, JIAO W. A developer recommendation framework in software crowdsourcing development [C]//National Software Application Conference. Berlin: Springer, 2016: 151-164.

[6] ZHU J, SHEN B, HU F. A learning to rank framework for developer recommendation in software crowdsourcing[C]// 2015 Asia-Pacific Software Engineering Conference. Piscataway: IEEE, 2015: 285-292.

[7] ZHANG Z, SUN H, ZHANG H. Developer recommendation for Topcoder through a meta-learning based policy model[J]. Empirical Software Engineering, 2020, 25(1): 859-889.

[8] AUER S, BIZER C, KOBILAROV G, et al. Dbpedia: A nucleus for a web of open data[C]//The Semantic Web. Berlin: Springer, 2007: 722-735.

[9] WANG Z, SUN H, FU Y, et al. Recommending crowdsourced software developers in consideration of skill improvement[C]//2017 32nd IEEE/ACM International Conference on Automated Software Enginee-ring. Piscataway: IEEE, 2017: 717-722.

[10] FU Y, SUN H, YE L. Competition-aware task routing for contest based crowdsourced software development[C]// 2017 6th International Workshop on Software Mining. Piscataway: IEEE, 2017: 32-39.

[11] 于旭, 何亚东, 杜军威, 等. 一种结合显式特征和隐式特征的开发者混合推荐算法[J]. 软件学报, 2022, 33(5): 1635-1651.
YU X, HE Y D, DU J W, et al. Developer hybrid recom-

mendation algorithm based on combination of explicit features and implicit features[J]. Journal of Software, 2022, 33(5): 1635-1651. (in Chinese)

[12] 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. New York: ACM, 2016: 1-10.

[13] ZHANG W, ZHAO J, PENG R, et al. SusRec: An approach to sustainable developer recommendation for bug resolution using multimodal ensemble learning[J]. IEEE Transactions on Reliability, 2022, 72(1): 61-78.

[14] WANG H, ZHANG F, WANG J, et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems[C]//Proceedings of the 27th ACM International Conference on Information and Knowledge Management. New York: ACM, 2018: 417-426.

[15] QU Y, BAI T, ZHANG W, et al. An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation[C]//Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data. New York: ACM, 2019: 1-9.

[16] YANG D, GUO Z, WANG Z, et al. A knowledge-enhanced deep recommendation framework incorporating gan-based models[C]//IEEE International Conference on Data Mining. Piscataway: IEEE, 2018: 1368-1373.

[17] WANG H, ZHANG F, ZHAO M, et al. Multi-task feature learning for knowledge graph enhanced recommendation [C]//The World Wide Web Conference. New York: ACM, 2019: 2000-2010.

[18] SHI C, ZHANG Z, LUO P, et al. Semantic path based personalized recommendation on weighted heterogeneous information networks[C]//Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. New York: ACM, 2015: 453-462.

[19] XIAN Y, FU Z, MUTHUKRISHNAN S, et al. Reinforcement knowledge graph reasoning for explainable recommendation[C]//Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, 2019: 285-294.

[20] WANG X, HE X, CAO Y, et al. Kgat: Knowledge graph attention network for recommendation[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM, 2019: 950-958.

[21] BORDES A, USUNIER N, GARCIA-DURAN A, et al. Translating embeddings for modeling multi-relational data [C]//Proceedings of the 26th International Conference on Neural Information Processing Systems. Red Hook: Curran Associates Inc, 2013: 2787-2795.

[22] WANG Z, ZHANG J, FENG J, et al. Knowledge graph embedding by translating on hyperplanes[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Menlo Park: AAAI Press, 2014: 1112-1119.

[23] LIN Y, LIU Z, SUN M, et al. Learning entity and relation embeddings for knowledge graph completion[C]//Proceedings of the AAAI Conference on Artificial Intelligence. Menlo Park: AAAI Press, 2015: 2181-2187.

[24] JI G, HE S, XU L, et al. Knowledge graph embedding via dynamic mapping matrix[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Stroudsburg: ACL, 2015: 687-696.

[25] HUANG X, ZHANG J, LI D, et al. Knowledge graph embedding based question answering[C]//Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. New York: ACM, 2019: 105-113.

[26] HAKKANI-TÜR D, CELIKYILMAZ A, HECK L, et al. Probabilistic enrichment of knowledge graph entities for relation detection in conversational understanding[C]//Interspeech 2014. Red Hook: Curran Associates, Inc, 2014: 2113-2117.

[27] AGARWAL D, CHEN B C. Regression-based latent factor models[C]//Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2009: 19-28.

[28] KIM Y. Convolutional neural networks for sentence classification[C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2014: 1746-1751.

[29] ZHENG L, NOROOZI V, YU P S. Joint deep modeling of users and items using reviews for recommendation[C]// Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. New York: ACM, 2017: 425-434.

[30] MIKOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[C]//Proceedings of the 26th International Conference on Neural Information Processing Systems. Red Hook: Curran Associates Inc, 2013: 3111-3119.

[31] LIU D, HE M, LUO J, et al. User-event graph embedding learning for context-aware recommendation[C]//Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. New York: ACM, 2022: 1051-1059.
[32] SCHLICHTKRULL M, KIPF T N, BLOEM P, et al. Modeling relational data with graph convolutional networks[C]//European Semantic Web Conference. Cham: Springer, 2018: 593-607.
[33] RENDLE S. Factorization machines with libfm[J]. ACM Transactions on Intelligent Systems and Technology, 2012, 3(3): 1-22.

作者简介

杜军威　男,1974年7月出生,山东威海人. 博士,教授,CCF高级会员,博士生导师. 主要研究方向为智能软件工程、推荐算法和自然语言处理.
E-mail: djwqd@163.com

王昭哲　男,1997年1月出生,河南商丘人. 现为硕士研究生. 主要研究方向为推荐系统.
E-mail: wzz9701@163.com

于　旭(通讯作者)　男,1982年7月出生,山东青岛人. 博士,教授,CCF高级会员. 主要研究推荐算法、迁移学习、智能软件工程.
E-mail: yuxu0532@163.com

胡　强　男,1980年6月出生,山东邹城人. 博士,副教授, 硕士生导师. 主要研究方向为服务计算、人工智能.
Email: huqiang200280@163.com

江　峰　男,1978年10月出生,江西彭泽人. 博士,教授,CCF专业会员. 主要研究领域为机器学习,缺陷预测.
E-mail: jiangkong2002@163.com

巩敦卫　男,1970年3月出生,江苏徐州人. 博士,教授,CCF杰出会员. 主要研究智能优化与控制、基于搜索的软件工程.
E-mail: dwgong@vip.163.com