基于多开发者社区的用户推荐算法*

时宇岑[1], 印莹[1], 赵宇海[1], 张斌[1], 王国仁[2]

[1](东北大学 计算机科学与工程学院,辽宁 沈阳 110169)

[2](北京理工大学 计算机学院,北京 100000)

通讯作者：印莹,E-mail: yinying@ mail.neu.edu.cn

摘 要: 互联网技术迅猛发展。基于开发者社区的提问-回答经验交流方式已成为众多开发人员解决软件开发、维护过程中所遇问题的重要手段之一。如何为开发者社区中的提问者及时、准确地推荐问题回答者是具有实际需求的重要问题。本文通过对 Stack Overflow 和 Github 两个具有代表性的主流开发者社区相关数据的收集和分析，观察到影响上述问题推荐准确性和反馈及时性的三个现象：(1) 用户标签自定现象，即开发者社区中，用户的标签信息是由用户自己主观定义所得，而非系统根据用户的历史行为客观标定；(2) 不对称活跃现象，即用户可能在某个或某些开发者社区中活跃，但在其它社区中并不具有同等活跃程度，甚至不活跃；(3) 关键词集封闭现象，即开发者社区中的问题回答者推荐仅依据问题文本中的关键词，而未考虑其它语义相关的关键词。针对以上问题，本文融合开发者社区的用户信息，通过分析用户与用户之间的互动行为，建立跨社区的开发者网络，并提出一种基于重启随机游走的算法更新用户标签。进一步，通过使用 Taxonomy 来扩充问题的查询关键词范围，在此基础上，协同用户矩阵进行更加准确的推荐，并增大了推荐时有效用户的范围。本文收集的实验数据包括 170 万有效主题，累计 40 万用户以及 117 个标签。实验结果证实，提出的算法具有较好的 F-measure 和 NDCG 度量。特别是，在冷门标签的推荐中，与未采用本文方法的推荐算法相比，基于 NDCG 度量的推荐准确率至少可提高 2 倍，部分甚至可高达 4 倍。

关键词: 多开发者社区，重启随机游走，Taxonomy，协同过滤，推荐系统

A User Recommendation Algorithm based on Multi-developer community

Shi Yu-Cen[1], Yin Ying[1], Zhao Yu-Hai[1], Zhang Bin[1]，Wang Guo-Ren[2]

[1](School of Computer Science and Technology, Northeastrn University, Liaoning 110169, China)

[2](School of Computer, Beijing Institute of Technology 100000, China)

Abstract: Internet technology is developing rapidly. Based on the developer community's question-answering experience communication method has become one of the important means for many developers to solve problems encountered in software development and maintenance. How to promptly and accurately recommend a question responder to a questioner in the developer community is an important issue with practical needs. Through the collection and analysis of the data of two representative mainstream developers in Stack Overflow and Github,we observe three phenomena that affect the timeliness and accuracy of the above recommended questions: (1) User label customization phenomenon: In the developer community, the user's tag information is subjectively defined by the user, rather than the system is objectively calibrated according to the user's historical behavior; (2) Asymmetric activity：the user may be active in one or some developer communities However, it is not equally active or even inactive in other communities; (3) Keyword set closure phenomenon, that is, the question answerer in the developer community recommends only based on the keywords in the question text, but does not consider other semantic related Key words. In view of the above problems, we integrate the user information of the developer community, analyzes the interaction between users and users, establishes a cross-community developer network, and proposes an algorithm based on restart random walk to update user tags. Further, by using Taxonomy to expand the query keyword range of the problem, on the basis of this, the user matrix is more accurately recommended, and the range of effective users at the time of recommendation is increased. Finally, the experimental results of F-measure and NDCG are good, which can effectively improve the efficiency and accuracy of problem recommendation.

Key words: Multi-developer community, restart random walk, Taxonomy, collaborative filtering, recommendation system

---

基金项目：国家自然科学基金项目面上项目 61772124；国家自然科学基金重点项目 61332014；中央高校基本科研业务费项目（N150402002)

Foundation item:the National Natural Science Foundation of China (61772124);the State Key Program of National Natural Science of China (61332014);the Fundamental Research Funds for the Central Universities (150402002).

互联网技术的迅猛发展深刻变革了许多领域的传统认知方式。例如，在软件工程领域，基于开发者社区的提问-回答经验交流方式已被众多 IT 工作者广泛接受，成为解决软件开发、维护过程中所遇问题的重要手段之一。如何为开发者社区中的提问者推荐能及时、准确解决问题的回复者是一个具有实际需求的重要问题。本文通过对 Stack Overflow 和 Github 两个具有代表性的主流开发者社区大量相关数据的收集和分析，观察到影响上述推荐问题准确性和反馈及时性的三个现象：

（1）用户标签自定现象：通过对 Stack Overflow 和 Github 用户注册过程中标签选择及其后续变化情况的分析调研发现，开发者社区中用户标签信息是由用户主观自定的。因此，存在着两种情况：①用户未选定某标签，但参与了该标签对应的问答活动；②用户选定了某标签，但实际并未参与该标签的问答活动。这两种情况都会影响推荐的准确性和反馈的及时性，其根源就在于用户标签的主观自定。

（2）不对称活跃现象：通过对 Stack Overflow 和 Github 相关数据的搜集和分析，发现存在着部分用户在一个社区（如 Stack Overflow）中十分活跃，参与大量问题回答，而在另一个社区（如 Github）中极少参与问题回答。例如，图 1 中给出了用户“Parvin Gasimzade”在两个不同社区中的表现。由图 1(a)可见，该用户在 Github 中的 Repositories 值为 3，代表其在 Github 中仅参与了 3 次问答活动，而在 Stack Overflow 中用户的 POSTS 值为 143，代表该用户仅在 mongdb 标签下就参与了 143 次问答活动，其对应的 SCORE 值为 846，代表该用户具有 846 分的正向评分。很明显，该用户在 Github 中参与问答活动的活跃度明显低于其参与 Stack Overflow 社区问答活动的活跃度。调研发现，与用户“Parvin Gasimzade”情况类似的其他用户并不在少数。

(a) The profile of user "Parvin Gasimzade" in Github　　(b) The profile of user "Parvin Gasimzade" in Stack Overflow

(a) 用户“Parvin Gasimzade”在 Github 中的行为　　(b) 用户“Parvin Gasimzade”在 Stack Overflow 中的行为

Fig1 The profiles of user "Parvin Gasimzade" in two communities

图 1 用户“Parvin Gasimzade”在两个社区中的表现

（3）关键词集封闭现象：软件开发人员在调试遇到的问题时，经常会将问题的报错信息直接复制粘贴到开发者网站中的搜索框内进行查询。Stack Overflow 和 Github 中的推荐算法通过提取用户提交文本中的关键词，利用关键词匹配进行推荐。推荐结果仅根据用户提交文本中获取的关键词给出，忽略了其他语义相关关键词对推荐结果的有效扩充。封闭的关键词集可能导致有效推荐结果被忽略，影响推荐准确性和反馈及时性。

开发者社区作为当前软件开发人员面临问题时寻找解决方案的主要参考平台，在软件开发领域有着举足轻重的地位[6]。因此，针对开发中存在的各种问题，对于推荐算法进行改良可以有效提高开发者社区中的推荐效率。针对现象 1，通过一种基于重启随机游走的算法对开发者网络中的用户标签进行更新，有效降低了用户标签的主观性。针对现象 2，提出一种多开发者社区的网络构建方法，将不同社区中的开发者进行链接，建立跨开发者网络以提高问题的推荐效率。针对现象 3，使用 Taxonomy 对问题的标签范围进行拓展，进一步联合构建的开发者网络进行最后的用户推荐排名。大量实验评估表明，本文所建立的开发者网络有较好的准确性且可以有效增强所获的用户与标签间关系的客观性，通过使用 Taxonomy 也可以有效提高问题推荐的准确度以及反馈及时性。

本文余下部分的组织结构如下：第 1 节对问题以及相关定义进行描述。第 2 节具体阐述本方法的设计思想与具体实现。第 3 节描述整个实验的设计和实施，给出实验结果并进行分析。第 4 节概述本文的相关工作。最后，第 5 部分对全文进行工作总结与展望。

1 问题定义及研究思路

1.1 相关定义

本文的相关定义是关于第二节中识别用户应该用到的文本相似度计算，以及计算用户与用户关系所应用到的线性归一化定义。

定义 1(文本相似度).文本相似度主要是针对两个标签间的相似度进行衡量。因此本文对于标签 a 与标签 b 进行文本相似度计算 *sim*(*a*,*b*)由公式 1 定义如下：

$$sim(a,b) = \frac{a与b中最长公共子串长度}{a与b最长短语的长度} \tag{1}$$

定义 2(线性归一化).对于一个集合 A 内的所有元素 a_i(i<=N)进行线性归一化处理后的每个元素数值是通过其原本数值与集合内所有元素数值之和相除所得，集合 A 中的元素 a_i'的线性归一化计算如公式 2 所示：

$$a_i' = \frac{a_i}{\sum_{k=1}^{N} a_k} \tag{2}$$

1.2 问题描述

本文跨社区网络构建主要是将用户与用户，用户与标签两部分信息构建成两个矩阵。其中，用户-用户矩阵为 n*n 阶矩阵，用户-标签矩阵为 n*m 阶矩阵。利用用户与用户和用户与标签的关系，在重启随机行走中通过矩阵间的运算得到更为准确的可以代表用户与标签关系的 n*m 阶用户-标签矩阵。

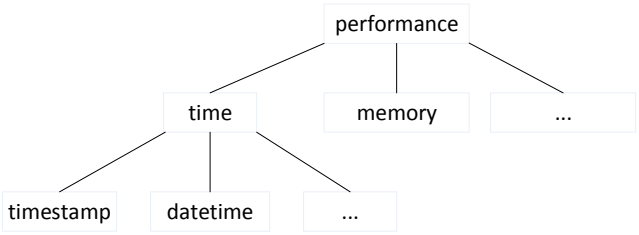

Fig2 An example of a Taxonomy tree rooted at performance

图 2 一个以 performance 为根的 Taxonomy 树的例子

受文献[15]中提出的 Taxonomy 启发，在得到更新的用户与标签关系后，建立如图 2 所示的具有上下义关系的标签树，用于开发者社区中的问题推荐范围拓展。最后，协同更新后的用户与标签关系进行问题推荐及用户权重计算，得出最终用户推荐排名。

1.3 研究思路

本文设计并实现了基于重启随机游走的用户标签更新算法与基于 Taxonomy 的用户推荐算法，主要目的是提高用户标签的准确度和问题推荐时的准确度。基于重启随机游走算法通过利用用户与用户之间的关系对用户与标签之间的关系进行更新，获得相比用户自我标记更加准确的用户偏好。基于 Taxonomy 的用户推荐算法在扩大问题推荐时可以匹配到的标签范围。方法整体框架如图 3 所示。

本文首先处理开发者社区中的用户数据，进行跨社区用户识别将相同用户合并后建立跨社区开发者网络。通过本文设计的重启随机游走算法更新用户与标签关系获得更加准确的用户标签信息。在此同时将开发者社区中的问题提取关键词并进行基于 Taxonomy 的问题内容拓展，进一步联合获得的用户标签信息进行权重计算最后根据权重排名进行用户推荐。

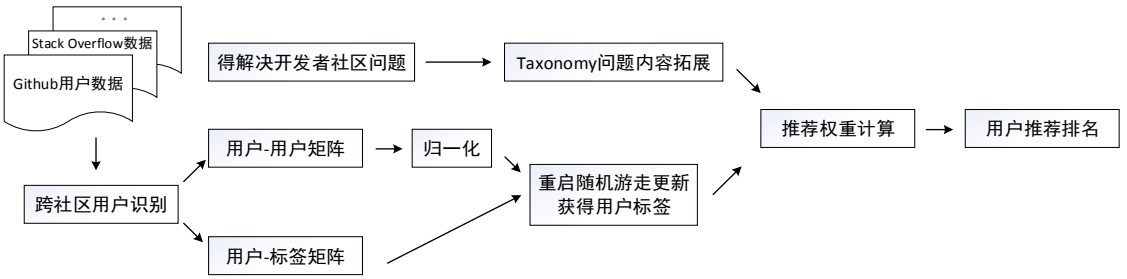

Fig3 The Algorithm framework

图 3 算法总体框架

2 基于多开发者社区的用户推荐算法

本文算法分为三部分进行工作：(1) 数据预处理，包括网络数据的收集处理以及跨社区相同用户的识别；(2) 用户标签更新，本文受到重启随机游走算法的启发，基于设计了一个基于重启随机游走的用户标签更新算法；(3) 用户推荐，本文首先使用工具 Taxonomy 对开发者社区中的标签建立语义树，通过语义树对开发者社区中的问题标签进行拓展，最后协同获得的用户标签关系进行用户推荐排名。

2.1 跨社区相同用户识别

通过统计不同社区中用户参与问题回答行为，本文得到用户在回答问题种类上的偏好并以用户与标签的关系表示。本文通过分析两个社区中名字相似的用户在不同问题上的偏好来识别不同社区中的相同用户。对于计算两个社区中用户的相似度 sim(a,b)，本文主要通过计算用户名称相似度 $sim_n(a,b)$与用户标签相同数量 ω 来获得，由定义 1 可得公式(3)。

$$sim_n(\mathrm{a},\mathrm{b}) = \frac{|LCS(seq(a),seq(b)|}{|\max(seq(a),seq(b))|} \tag{3}$$

其中，LCS()为计算用户 a 与用户 b 名字最长公共子串的长度，通过与二者中较长子串长度进行相除得到用户名称相似度。这里在衡量用户名称相似度时，与计算文本间相似度不同的是，本文未使用计算不用户名称之间的最长公共子序列而使用二者间最长公共子串。因为通过分析网络中用户在设定用户名称时，在首要期望的名称已被其他用户占用或者无法使用自己期望的特定字符的情况下，很大的可能会以用户名称加上一定的后缀或者插入一定的字符作为用户名进行设定。如果仅对于二者的公共子序列进行比较，在某些情况下可以寻找到相比寻找公共子串更相近的用户名称，但是该方法忽略了用户名本身字符的连接，很可能在相似度高的情况下仍然识别两个完全不相关的用户。并且在识别表现相近的情况下，计算公共子序列的开销要远大于计算公共子串。

在获取用户名称之间相似度之后，通过与用户间相同标签数量以一定比例进行相加，可以得到最终用户与用户之间的相似度 sim(a,b)。通过计算 sim()并设立一定的阈值，本文获得不同开发者社区中可能为同一用户的个人并进行分析。本文在识别用户时发现忽略空格以及字符大小写情况下，两个社区中包含大量名称完全相同的用户。但是，通过访问用户个人主页本文发现，虽然在名字完全相同且用户信息完整的情况下二者是同一用户的可能性很高，但仍然存在着两个社区的两位用户有着完全不同的兴趣偏好，本文认为不具有相同偏好的用户不可能成为不同社区中的相同用户。因此，在名称相似度作为识别不用社区中相同用户的条件下，本文引入了通过对比两个用户的标签进一步作为识别用户的参考条件。如公式(4)所示，在识别用户时，如果两位用户不具有相同的标签，二者的相似度直接设置为 0。在二者具有相同的标签的情况下，对二者名称相似度以及相同标签数量的权重以一定的形式相加获得最终二位用户的相似度。本文结合用户自己标记与本文统计的用户标签作为该用户的代表偏好，在保证对比相似度不大于 1 的情况下，在对权重计算时对其进行

取对数计算以减小相同标签数目过多对相似度判断造成的影响，则有：

$$sim(\mathrm{a},\mathrm{b}) = \begin{cases} 0 & \omega = 0 \\ sim_n(\mathrm{a},\mathrm{b}) + log_\alpha \omega & \omega \neq 0 \end{cases} \tag{4}$$

其中 α 代表在计算用户间相似度时保证名称相似度时，为了保证识别相同用户准确度而设定的用户相同标签数量系数，通过之后实验设定不同 α 最终取识别用户效果最好的 α 值。ω 代表识别用户该参考用户在两个社区间相同的标签数量。由于包含相同子串的用户数量可能不止一位，因此本文首先通过获得具有大于一定数值的用户相似度的用户进行识别，在之后保证识别准确度的前提下调整获取相同用户间计算 sim()的阈值来识别更多的用户。

2.2 基于重启随机游走的用户更新算法

在进行重启随机游走更新前，本文首先建立两个代表用户与用户关系和用户与标签关系的矩阵。用户与用户关系矩阵中每个元素代表每两位用户在开发者社区中的关系，因此用户与用户矩阵中元素 r_{ij} 可以通过进行用户之间关系权重叠加进行计算，公式 5 所示。权重计算中 n_1 代表用户 i 作为发问者发表主题且用户 j 参与该主题的主题数量，n_2 代表二者同时作为参与者在一个主题的主题数量即二者均不为发问者，I 为所有主题总数，α 代表问题发问者对于其他用户的影响力。在之后的实验中 α 取值为 0.66，即发布主题者对于参与主题的人权重加成是二者同时参与一个问题的权重加成的二倍。并且在实验中我们也发现这种设定所获得结果要优于两种权重直接相加。

$$r_{ij} = \begin{cases} \alpha \sum_{a=1}^{n_1} 1 + (1-\alpha) \sum_{b=1}^{n_2} 1 & i \neq j \\ 0 & i = j \end{cases} \qquad (0 \leq n1 \leq \mathrm{I}\ , 0 \leq n2 \leq \mathrm{I}) \tag{5}$$

但是在后续实验中本文发现在利用用户与用户之间的关系对用户与标签之间的关系进行更新时，存在部分关系系数较大的用户在更新时使一些用户的标签权重数值过大，在推荐时造成极大的影响。因此在统计用户关系权重之后，本文对于用户与用户之间的关系 r_{ij} 进行归一化处理，由定义 2 可得（公式 6）以防止上述情况的发生。至此可以获得用户与用户关系矩阵。

$$r'_{ij} = \frac{r_{ij}}{\sum_{k=1}^{N} r_{ik}} \tag{6}$$

与此相似，构建用户与标签矩阵需要对用户在开发者社区中问题的回答数据进行采集。用户对于标签的权重计算公式如公式 7 所示。其中，$s1$、$s2$ 分别代表用户在 Github 与在 Stack Overflow 中对于标签 j 的偏好次数，T 代表两个开发者社区中所有的标签数目，β 代表 Stack Overflow 中用户对于标签喜好的权重。由于用户与标签之间的关系直接作为推荐时的参考信息，不需要进行归一化处理。至此获得用户与标签关系矩阵。

$$d_{ij} = \beta \sum_{a=1}^{s1} 1 + (1-\beta) \sum_{b=1}^{s2} 1 \qquad (0 \leq s1 \leq T, 0 \leq s2 \leq T) \tag{7}$$

受到重启随机游走算法的启发，本文设计了一个利用用户与用户之间的关系对用户与标签之间的关系进行更新的算法。重启随机游走本质是从图中的某一个节点出发，每一步面临两个选择，随机选择相邻节点，或者返回开始节点。重启随机游走算法设定参数 a 为重启概率，1-a 代表移动到相邻节点的概率，经过一定次数的迭代到达平稳，平稳后得到的概率分布可被视作受开始节点影响的分布。而利用重启随机游走更新所获结果可以视作受到用户与用户之间关系影响得到的用户与标签之间的关系。本文将矩阵 M_T 视为最原始的用户-标签关系矩阵，在一系列迭代过程中，不断对 M_T 进行更新，最后获得的 M_T^n 作为最终获得的用户-标签矩阵。在每次迭代中本文将第 n 次迭代下用户与标签之间的关系 d_{ij} 与矩阵 M_O^n 用 d_{ij}^n 来表示，r_{ik} 与 d_{kj} 代表迭代时矩阵内的元素。因此，d_{ij}^{n+1} 本文可以用如下的公式(8)进行计算：

$$d_{ij}^{n+1} = \sum_{u_i \in p(u_k)} r_{ik} \cdot d_{kj}^n \tag{8}$$

其中 p(u_k)为一个返回用户 u_k 所指向的所有的用户的函数。为了保证迭代过程的收敛，本文引用了一个随机重启变量 c 与重启条件下的矩阵 M_P，则整个重启随机游走过程可以用如下公式(9)进行表达。其中重启矩阵 M_P 是由初始矩阵 M_O^0 得出。

$$M_O^{n+1} = c * M_u * M_O^n \ + (1-c) * M_P \tag{9}$$

当整个随机游走过程收敛到一个平稳状态时，可以获得一个稳定的用户-标签关系分布矩阵，其中的用户与标签之间的关系即为本文所需求最终目标，即一个概率分布矩阵 M_O^{n+1}。本文认为当所有元素满足公式(10)条件时，整个过程即为收敛。

$$\left|\frac{a_{ij}^n - a_{ij}^{n-1}}{a_{ij}^{n-1}}\right| \ \leq \ \varepsilon \ , i \in [1, N] \ , j \in [1, |T|] \tag{10}$$

通过控制收敛率可以控制重启随机游走进行的次数。通过设置不同的重启率，可以获得多种情况下使用重启游走更新后的用户-标签关系。

2.3 基于Taxonomy的用户推荐算法

在搜索内容扩充处理部分，本文使用了分类工具 Taxonomy。Taxonomy 是一种简化的知识库，它在表示标签之间的上下义关系时省略了许多节点之间的复杂的语义关系。Taxonomy 在结构上是一个多根树，在本文中，每一个节点代表一个标签，而每个标签之下也会有一个或多个子节点作为其子标签。例如，标签“Scrapy”在 Taxonomy 即为标签“Python”的一个子节点。通过 Taxonomy 的扩展可以有效地扩大问题标签的命中范围与权值的准确度。

算法 1 搜索内容扩充算法：

Input：**T = {t_1 , t_2 , t_3 … t_m}**

Output：**T_q = { (t_1', w_1) , (t_2', w_2) , (t_3', w_3) … (t_n', w_n) }**

1: for all t_i ∈ **T** do　　//只提取存在于开发者社区的关键词

2:　　if t_i ∈ **T** then

3:　　T_q ← Add(t_i,1)　　//加入本身

4:　　t_p ← GetParent(t_i)

5:　　if t_p **exists** then

6:　　T_q ← Add(t_p , c)　　//加入父亲标签（如果存在）

7:　　end if

8:　　for all t_b ∈ GetBrother(t_i) do

9:　　T_q ← Add(t_b, c)　　//加入兄弟标签（如果存在）

10:　　end for

11: end if

12: end for

13: return T_q

通过获得的 Taxonomy 树，本文设计了如算法 1 所示的搜索内容扩充算法。搜索内容扩充算法是对于本文提取出的所有单词 T，保存所有在本文所提取的标签中出现的部分。如果问题文本中提出单词{this，is，java，spl}，T 作为输入取开发者社区中存在标签即为{java，sql}。对于所有 T 中的节点，都会进行如下处理。在确定原有单词标签的权重前提下，对于有父辈节点的单词，将其父辈节点以及一个关系系数 c_1 加入输出 T_q 中。同理，之后对于有兄弟节点的单词，本文将其兄弟节点以及一个关系系数 c_2 加入输出 T_q 中。通常情况下 c_1 与 c_2 可以取相同的系数以简化算法。其中 Add(t_i,a)函数是将权重 a 加进标签 t_i 在结果集的权重，GetParent(t_i)以及 GetBrother(t_i)分别获得当前节点 t_i 的父亲节点和兄弟节点，权重的计算方法我们沿用了文献[15]中的设置。最后算法输出的 T_q 中 t_n 代表原标签以及根据 Taxonomy 新加入的标签，每个 t_n 后的 w_n 代表该标签经过算法 1 处理后所占权重。

用户排名算法如算法 2 所示。其中 T_q 代表本文提取并引入后每个关键词/标签的权重，T 代表更新前用户

对于单个问题所打的标签，U_t 为本文更新后用户与标签之间的关系向量， w_n 代表每个用户对于各个标签的偏好的向量，为了保证两部分权重的平衡，此处在权重计算中 ω_1 和 ω_2 取值均为 0.5。当然后续可以根据不同的情况对 ω 的数值进行调整。

算法 2 用户排名算法

Input ： $T_q = \{ (t_1', w_1) , (t_2', w_2) , (t_3', w_3) \dots (t_p', w_p) \}$， $T = \{t_1 , t_2 , t_3 \dots t_m\}$

$U_t = \{w_1,w_2,\dots,w_n\}$ $w_i = \{(r_1,a_1),(r_2,a_2),\dots,(r_n,a_n)\}$

Output ：用户排名 $L = \{r_1,r_2,\dots,r_n\}$

```
Initalize L = {r1,r2,…,rn} , Scores = {0,0,…,0}
for all ri∈R do
    for all tj∈Tq do
        if tj ∈ Ut then
            Scores1(ri) =Tqj * Uj          //以更新后的问题关键词信息计算评分
        end if
    end for
    for all tk ∈ T do
        if tk ∈ Ut then
            Scores2(ri) =Tq * Uk           //以更新前的问题关键词信息计算评分
        end if
    end for
end for
Scores(ri) = Scores1(ri) *ω1 + Scores2(ri)*ω2          //用户推荐权重计算
L ← Rank Score in descending order and select top-k
Return L
```

在用户推荐过程中，首先对用户排名数组初始化，在初始情况下每个用户所具有的推荐权重为 0。而推荐部分本文对两部分内容进行了参考，即使用 Taxonomy 更新后的关键词参考推荐，与使用未更新的用户直接标签的内容进行推荐，将两部分内容以一定权重求和作为推荐权重。对于使用 Taxonomy 更新后的关键词权重更新，本文通过计算关键词在文档中的权重与用户与标签间权重二者的乘积，获得每个用户对于文档的偏好权重。对于未更新的用户对于文档直接打标签部分，本文默认文档用户中所有手动标记标签权重均相等，在进行归一化处理后同用户与标签之间权重进行乘运算，获得未使用 Taxonomy 的用户标签推荐。二者在一定权重下（ω_1, ω_2）相加获得最终每个用户对于文档的偏好程度，在由高到低进行排名后通过网站进行推荐。其中 Scores()为通过计算用户标签向量与问题中标签权重向量进行乘积来获得的用户推荐积分的函数。

3 实验与分析

3.1 实验数据集

由于本文算法所需数据基于开发者社区中的用户行为，需要利用用户与用户之间的关系信息以及用户与标签之间的偏好信息进行实验。因此通过抓取开发者社区中的所有有效问题，以获得有意义的用户在开发者社区中的行为信息。本文使用 Python 中的工具 Scrapy 进行开发者网站中的数据抓取工作。首先收集开发者社区中的标签信息，通过标签进行初步筛选，爬取在本文标签分类下每类标签包括的问题信息。进一步，爬取每个问题下所有用户的参与行为，统计用户与用户和用户与标签的信息。最后在获得用户有效信息之后通过数个预处理方法对所获数据进行标签语义处理，包括根据问题评分进行有价值的主题筛选与有效用户分类等，得到用于实验的有效数据集。本文数据集如表 1 所示，本文收集数据包含截止到 2017 年 10 月的 Stack Overflow 以及 Github 两个社区中的共计 117 个标签下约 140 万有效主题贴。对所有主题贴统计参与其中的用户并排除无法作为推荐参考的游客账号以及类似 google 的大型公共账号，累计获得两个开发者社区中的约 40 万有效注册用户的 id 及其在有效问题下的活动信息。

Table1 Data set of our paper

表 1 本文数据集

	Stack Overflow	Github
标签数	618	117
主题(问题)数	1355700	311821
用户数	349355	48120
相同标签数	117	

3.2 用户识别实验

如图 4 所示为本文统计高度相似用户名称并进行用户识别的实验。前 2w 名作为活跃部分的用户中，我们统计出确定为不同社区间用户是否相同的占比率与 2w 名之后用户无太大区别，即所有用户活跃区间中，可以完全直接确定二者为相同用户的所占比例相同，并且在确定条件下二者具有至少一个标签相似且大部分情况为至少拥有两个相似标签。但由于非常活跃的用户本身所具有的标签数量很高，会造成在跨社区比较时确定二者为相同用户的概率增加，而具有相同标签数量较多但无法确定为相同用户的情况会减少。排除活跃用户普通用户与不活跃用户都在发现名称相近且具有相同标签但无法确认占有较高的比率。由于 Github 可以赋予用户的标签数量稀少，这可以通过图中无法确定的用户部分即深蓝色部分可以发现，大部分非活跃且无法确定的用户中具有一个相同标签的比重占很大部分。而对不同社区中可以确定的相同用户，仅有一个标签相同的情况则相对较少。因此本文在对用户识别时，将相似度阈值设为至少大于 1。即如果两位用户在不同社区具有相同的名称但标签数相同数目不大于 1 则不将他们识别为同一开发者，这样可能会损失大约 2%现实可以确定为同一用户的跨社区人员。

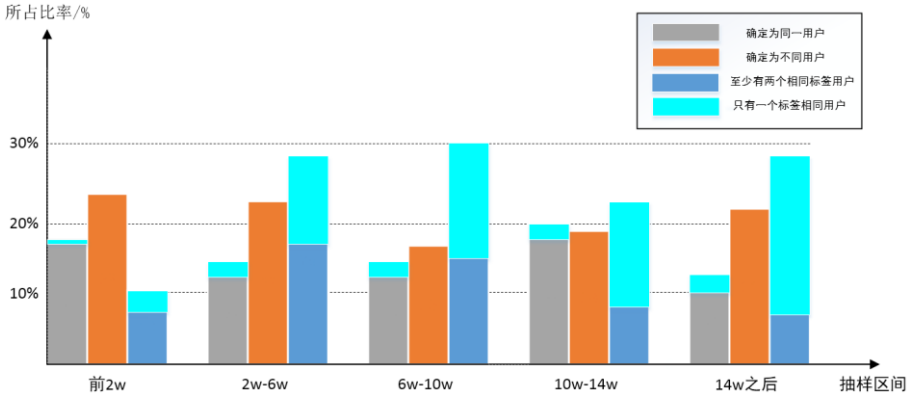

Fig4 Overall ratio of three users under sampling conditions

图 4 抽样条件下的三种用户所占总体比率

识别用户的实验中不同标签选择表现基本相同，例如在标签 sql 下识别用户几种情况所占比率为 14%、21%和 25%。在阈值设定为 1 情况下，除了统计存在相似用户名称的用户占比，本文将标签热度进行分类，即标签具有主题大于 20w，大于 10w 以及大于 1w，并对 α 分别取值 2、3、4、5、6 和 7 进行实验获得识别用户数量的占比率结果如图 5 所示。可见对于两个社区在 α 取值大于 3 后识别用户数量占比趋于平稳，为本文算法可以识别出的相同用户数量占所有用户比例。

已有的跨社区用户识别算法多数是基于用户名称与主页之间的关系进行识别，与本文基于用户标签偏好的识别条件不同。部分算法通过实验可以识别出的跨社区用户数量约占实验用户数量的 20%。相对该算法本

文可以确定识别的用户数量已达到 15%，并且还有 10%以上高概率相似的用户可以作为相同用户参考，因此本文算法在保证识别准确度的前提下在识别用户的数量上也有着较好的表现。但相对本文的变量的取值适用于 Stack Overflow 以及 Github 两个社区，如果应用到其他社区需要对数据进行重新统计及变量的重新计算。

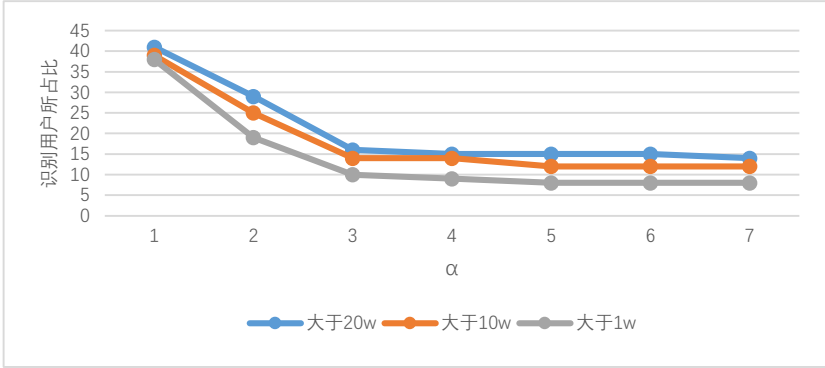

Fig5 Different alpha values and number of identified users in the label area

图 5 不同 α 取值与标签区域下的识别用户数量

3.3 用户标签更新实验

重启随机游走的重启率通常情况下设置为 0.75，重启率代表更新后的用户矩阵受原用户矩阵数值的影响，本文希望可以通过控调整重启率（restart）以获得更加准确的用户对于标签数量的偏好以及更加准确的用户对于标签偏好的权重。对于标签更新实验将标签按照当前开发者社区具有主题数量划分为 3 个区域，对 3 个不同活跃度标签分别进行 F-measure 的计算并取平均值作为该区域中的标签更新显著程度，其中三个区域分别为主题帖数量大于 10w 的热门标签约 50 个，主题帖数目大于 5w 的非热门标签约 100 个和剩下的所有主题帖数目大于 1w 的冷门标签，实验结果如图 6 所示。

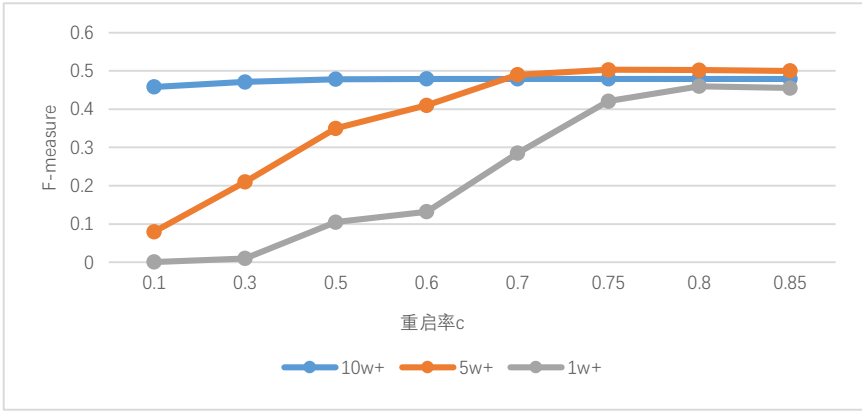

Fig6 Different label area F-measure values under different restart rates

图 6 不同重启率下的不同标签区域 F-measure 值

更新标签时热门标签中尤其是具有 100w 以上主题帖的标签在用户标签更新后无论重启值的多少，在迭代达到收敛条件后一定会被引入到每个用户的偏好下，根据重启值的增加新加入的标签权重也随之增加。虽然对用户标签更新会造成所有用户都具有热门标签偏好的情况，并且更新后原对标签具有较高权重的用户的偏好值也会下降。对热门标签问题下的用户推荐进行分析，如果问题中含有热门标签且热门标签在问题中的权重较高，则热门标签偏好值较低的用户几乎不会出现在推荐用户中。因为本文是根据用户适合问题的权重

来进行用户推荐，而新加入标签与更新用户标签权重的主要功能分别是为冷门问题提供更多的用户以及推荐用户时平衡用户排名权重大小。对在某个领域具有较高地位的用户进行权重更新则对于该用户的领域地位不会造成太大影响。

3.4 用户推荐实验

3.4.1 NDCG

NDCG（归一化折损累积增益）允许以实数形式对排名进行相关性打分。本文中 NDCG 是由通过排名结果计算 DCG 值得出的，二者的计算公式如下公式 11 与公式 12 所示：

$$DCG@K = \sum_{i=1}^{k} \frac{2^{r(l)}-1}{log_2(i+1)} \quad (11)$$

$$NDCG@K = \frac{DCG}{iDCG} \quad (12)$$

两个公式中 K 代表推荐的列表长度即推荐用户数量。公式 11 中 DCG 为通过本文算法得到的用户排名后推荐用户的相关性。公式 12 中 iDCG 为理想状态下的 DCG 的值，iDCG 的计算是本文对实验选择的问题使用原始用户标签权重衡量用户与问题的相关性并添加标记。即 DCG 计算中，r(l)代表该用户在推荐时的相关等级。由于推荐用户时可以被本文寻找的用户通常情况下都为活跃用户，因此根据用户本文，在本文中 r(l)的取值分别为 2：非常适合，1：适合与 0：不适合。这里不包括-1 的错误推荐的情况因为可以推荐出的用户一定会对问题中的一个或多个标签进行过问题回答，因此只能判断这位用户适不适合于回答这个问题。并且在之后的实验中，对于 log2(i+1)中 i 的取值，本文分为两种情况即首先计算所有用户的价值均相等的情况来分析本文抽取一定数量下用户的准确度，以及根据用户排名用户价值递减的分析推荐用户的排名的准确度。

3.4.2 实验结果

用户的推荐方面本文选取 Stack Overflow 中不同热门程度下的标签作为当前问题的最重要标签进行推荐实验。我们在实验中将标签以热度划分为 3 类，并且每类标签之间的表现在实验中基本相同。

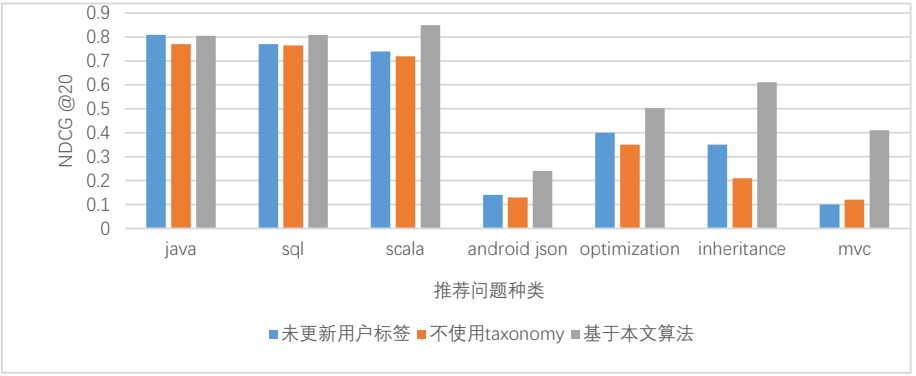

Fig7 Consider user's recommended performance under different master tags under the ranking weights

图 7 考虑排名权重下不同主标签下用户推荐表现

如图 7 所示，实验中本文分别选取热门标签下编程领域中的几个常用标签：java，sql 与 scala 以及冷门标签下的 android json。在其他冷门标签领域中，本文选取了“编程表现”下的两个标签：optimization 与 inheritance 和问题以概念为主的 mvc 标签。实验中选择将这些标签作为主要标签的问题，即问题在提取标签权重后该标签的权重数值最高，代表问题与主要标签的相关度最高，每个问题也包含除主要标签的其他标签。实验包含了两个社区中的热门和冷门以及三个不同领域下的标签，可以代表本文算法在两个社区中的不同的热度领域以及推荐方向领域推荐的表现。实验结果如图 7 所示。在热门标签下，使用 Taxonomy 对于推荐准确度的提升有限以至不同抽样下推荐的准确度在部分标签下提升不明显。因为提出问题时对于热门标签选择偏好的人员

远多于冷门标签，活跃人员对于标签的偏好权重也很高。本文在推荐用户时发现，在进行 java 等热门标签推荐时使用三种推荐方法所获得的用户排名大致相同，仅推荐用户时根据用户的权重排名有微小的变化。在本文算法下，由于用户对于标签的权重经过计算后过热门标签下的偏好的数值通常会减少，因此使用本文方法推荐用户之间权重相差值较小，相对于参考未使用本文策略算法更新的用户偏好矩阵进行推荐的用户之间差值不明显。但结果上二者推荐的用户排名结果大致相同。因此本文也进行了不考虑用户之间推荐权重的实验，假定选中的前 20 名用户所代表的推荐权重一样，即在 NDCG 的计算之中 $\log_2(i+1)$中 i 恒定为 1，结果如图 8 所示。

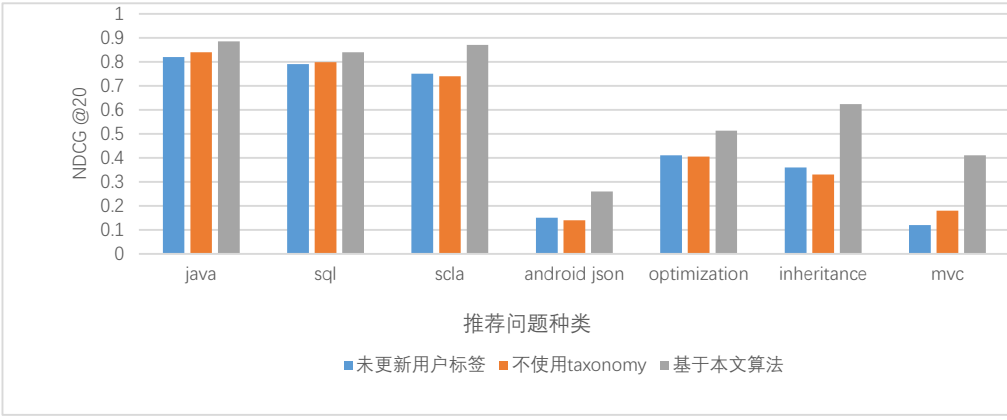

Fig8 Do not consider the NDCG value under the user's ranking weight

图 8 不考虑用户排名权重下的 NDCG 值

可以看出在不考虑用户权重情况下，本文算法进行用户推荐的准确值有一定程度的增加，说明在问题属于热门标签领域的情况下，本文算法和普通推荐结果之间差距较小但也存在至少 3%的提升率，部分热门标签下可以达到 10%。并且本文算法对于推荐一定数量用户进行回答的表现要相对使用普通条件推荐表现更加准确。即在于冷门标签的推荐情况，通过搜索内容扩充可以将问题的描述标签数量增加并且也扩展了用户的标签数量，相对于未使用 Taxonomy 的条件进行推荐，明显可以推荐更多数量且更加适合回答问题的用户。非热门标签领域下提升可以达到 10%以上，部分冷门标签的提升率可以达到 26%。对于未更新的问题用户标签种类，单个用户活跃的标签数量较少造成了冷门标签对应的用户数量稀少。加入对于用户标签关系的更新，可以扩大单个用户在推荐时命中相应领域的范围，进而扩大推荐时可以参考的用户数量。虽然当前研究环境下热门标签推荐已经达到一定的准确度，但是本文算法在热门标签领域上的推荐准确度仍然有一定的提升。并且根据本文统计热门标签数量仅占所有实验标签的 8%，而本文的算法可以在其余非热门标签领域下进行更为高效的问题推荐。

4 相关工作

本章将对与本文工作相关的用户识别，重启随机游走以及Taxonomy分如下3节进行介绍。

4.1 用户识别

为了识别不同开发者社区中的同一用户，当前已有如下相关研究。Goeminne 等人[1]将社区中的每个用户表示为一对二维向量（用户名，电子邮件）。他们提出如果两对向量共享至少一个元素或者至少一个共享元素对具有大于某个阈值 minLen 长度的相同子串，则他们是同一个人。之后，Bird 等人[2]提出了更先进的算法。他们没有仅仅定义一些简单的匹配规则，而是首先测量文本间的 Levenshtein 距离，Levenshtein 是一个用来衡量两个用户名之间的相似性的参数。然而，Goeminne 和 Bird 在研究中都没有考虑名字和电子邮件中的语义信

息。针对该问题，Erik[3]等人提出了一种基于语义的方法，通过应用潜在语义分析（LSA）来识别同一用户。

社交网络中的身份链接问题也是 Facebook 和 Twitter 等社交网络公司的热门研究课题。Iofciu 等人[4]将每个用户配置文件被表示为一个向量，通过计算两个向量之间的相似度来链接用户。另一方法将用户生成的内容（如推文）以及用户行为（如书写风格）也考虑在内。Liu 等人[5]提出了一个链接社会身份的框架 HYDRA。该方法考虑用户的异构行为，建立一致性矩阵来模拟不同社交网络中的结构一致性。但是实际上开发者社区中的用户识别与社交网络不同，人们在开发者社区中所能产生的内容的质量和类型远不及社交网络中的内容。例如，开发者社区的用户不允许共享他们的位置。另外，一些开发者社区在用户之间缺乏拓扑结构，如 Stack Overflow 中的用户间无法跟随。

4.2 重启随机游走

随机游走（Random walk)算法，是从一个或一系列顶点开始遍历一张图。在任意一个顶点，遍历者将以概率 1-*a* 游走到这个顶点的邻居顶点，以概率 *a* 随机跳跃到图中的任何一个顶点，称 *a* 为跳转发生概率，每次游走后得出一个概率分布，该概率分布刻画了图中每一个顶点被访问到的概率。用这个概率分布作为下一次游走的输入并反复迭代这一过程。当满足一定前提条件时，这个概率分布会趋于收敛。收敛后，即可以得到一个平稳的概率分布。重启随机游走算法是在随机游走算法的基础的改进。从图中的某一个节点出发，每一步面临两个选择，随机选择相邻节点，或者返回开始节点。算法包含一个参数 *a* 为重启概率，1-*a* 表示移动到相邻节点的概率，经过迭代到达平稳，平稳后得到的概率分布可被看作是受开始节点影响的分布。重启随机游走可以捕捉两个节点之间多方面的关系，捕捉图的整体结构信息。当前研究下重启随机游走的工作主要集中于查询、检索及算法的优化方面。Chao 等人[16]在查询方面提出了两种算法 Squeeze 和 Ripple，其中 Ripple 通过估计落在节点附近的重启随机游走分数采取相应策略，而不需要评估附近节点，在医疗查询，广告等方面有不错的表现。检索方面 Masaki 等人[17]提出了一种基于共引网络的文件推荐算法，通过解析引用文献的全文为共引网络中文件间的边设立权重，并使用重启随机游走应于该加权图中进行文档推荐。Yu 等[18]人提出了一种动态调度算法，可以有效优化动态更新时使用重启随机游走算法计算相似程度的开销。

4.3 Taxonomy

本文受到文献[15]中 Zhu 等人针对开发者社区 Stack Overflow 所提出的 Taxonomy 算法的启发，建立标签语法树协助用户推荐。Zhu 等人提取标签在开发者社区以及百科中的文本描述，通过这些文本分析标签之间的语义关系，建立多棵以某一标签为根节点且标签间具有父子关系的标签语义树，可以准确体现出开发者社区中标签间的关系。有关分类学（Taxonomy）的其他研究情况如下。基于百科的方法主要关注从百科中提取概念层次结构。Wiki Taxonomy[6]从维基百科分类系统构建分类。它包含 105,000 个包容关系，准确率为 88%。Kylin 等人[7]通过本体生成器（KOG）使用马尔可夫逻辑网络（MLN）来预测维基百科信息框类之间的包含关系。Yago 等人[8]将 Wikipedia 类别链接到 WordNet 同义词集。Yago 的实验中有超过 20 万个类别和 40 万个包容关系，准确度估计为 96%。但是开发者网站中的标签之间没有结构信息，难以应用基于百科的方法。基于网络的方法是在使用自由文本或社会标签的背景下提出的，Hearst 模式是最被广泛使用于基于自由文本的方法。Probase 等人[9]构建了最大的分类标准，其中包含 17 亿个网页中的 270 多万个类别。对于基于社会标签的方法，Mianwei Zhou 等人[10]引入了一个无监督模型来自动从社会注释中派生出层次语义。Jie Tang[11]等人提出了一种学习方法来捕捉标签的层次语义结构。Xiance Si[12]等人提出了三种方法来估计标签之间的条件概率，并使用贪婪算法消除冗余关系。Huai 等人[13]描述了一种从标签中提取本体结构的综合方法，该方法利用了低支持关联规则挖掘的功能，并辅以上层本体。WordNet.Zhishi.schema[14]是第一个从流行的中文网站的标签和类别发布一般分类标准的工作。这些传统基于标签的通用域方法仅使用注释文档来帮助包含检测。由百科描述开发者社区还包含其他信息，因此开发者社区更具有域特定性。传统的基于标签的方法可能不是最好的，因为开发者社区中的附加信息可能会增加分类结构的性能。

5 总结与展望

5.1 工作总结

前文详细介绍了本文的主要工作，包括数据集制作、跨社区网络建立以及用户标签更新与问题内容扩展等。以下，对全文工作加以总结。

（1）数据集制作。由于本文实验所需数据要求按标签将主题分类后处理用户信息，且需要获得主题的评分以判断主题是否具有实验价值，当前文献中提供的开源数据集无法满足本文对于实验数据的需求。因此，本文通过对开发者社区进行数据抓取，制作新的开发者社区用户信息数据集用于实验。全部数据集制作历经3个月的网站数据抓取及后处理工作，包括根据问题权重判断问题是否有效，过滤游客或者无效用户等。最后获得了包含117个标签下170万有效主题贴，累计40万以上用户的大规模真实实验数据集。

（2）跨社区网络建立。开发者社区中的问答推荐通常采取等待社区内会员回答或者推荐给内部用户进行回答的方式。因为存在用户标签自定、不对称活跃、关键词集封闭等现象，并采取回复等待的方式，这种方法可能会导致问题推荐后反馈不及时，局限性较大。针对该问题，本文提出了一种跨社区的问答推荐算法，通过建立跨社区的用户网络，将多个开发者社区中的用户连接起来，构建多个开发者社区中用户共存的网络进行问答推荐。在提高问题命中回复者的准确率同时，通过关键词集扩充，增大了推荐时有效用户的范围，使问题得到及时、准确回复的可能性大大增加。

（3）用户标签更新与问题内容扩展。本文设计的基于重启随机游走的用户标签更新算法与基于Taxonomy的用户推荐算法都是为了提高问题推荐的准确度以及扩大问题推荐的有效用户命中范围。基于重启随机游走的用户更新算法属于推荐系统中协同过滤算法的一种，通过使用用户与用户之间的关系，对用户与标签之间的权重进行一定程度的更新。由于用户对于自我的定位较为主观，而本文提出的算法可以分析用户在开发者社区中的表现，通过统计用户在各个领域下的问题回答参与情况和与其他用户的互动行为得出用户个人数据。之后，通过本文算法得到更为符合用户定位的用户标签偏好。对于开发者社区中的问题回复者推荐，仅依据文本中提取得到的关键词进行推荐可能会丢失有效的推荐用户。本文通过Taxonomy的语义树，对提取的关键词集进行拓展，利用拓展后的问题标签及其权重协同更新后的用户标签矩阵进行推荐。实验表明本文的方法对于推荐用户的数量提升明显，并且准确度也有较高的提升。

5.2 未来展望

本文是针对开发者社区中的几点现象，提出解决策略来提高开发者社区进行用户推荐效率以及准确度。在用户识别方面，本文对用户名称以及标签相似度提出一定的阈值，根据阈值来确定识别用户的范围。本文提出这种方法对于特定的两个社区适用性较强，但是仍有一定的局限性，将来可以在此算法基础上从更为复杂的用户之间的关系入手来识别不同开发者社区中的相同用户。而用户推荐方面还有推荐系统共同面临的冷启动、降噪等问题待解决，解决这些问题可以在本文基础上更多地提升推荐问题获得反馈的效率以及准确率。本文已经通过实验证明算法在开发者社区Stack Overflow以及Github中具有可实施性，下一步可以将该算法改良并推广到多个开发者社区中进行测试。

References:

[1] Goeminne M, Mens T. A comparison of identity merge algorithms for software repositories[J]. Science of Computer Programming, 2013, 78(8):971-986.athieu Goeminne and Tom Mens. A comparison of identity merge algorithmsfor software repositories. Science of Computer Programming,78(8):971–986, 2013.

[2] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and AnandSwaminathan. Mining email social networks. In Proceedings of the2006 international workshop on Mining software repositories[C], pages137–143. ACM, 2006.

[3] Erik Kouters, Bogdan Vasilescu, Alexander Serebrenik, and Mark GJvan den Brand. Who's who in gnome: Using lsa to merge softwarerepository identities. In Software Maintenance [C], 2012 28th IEEEInternational Conference on, pages 592–595. IEEE, 2012.

[4] Iofciu T, Fankhauser P, Abel F, et al. Identifying Users Across Social Tagging Systems[C]// International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July. DBLP, 2010.

[5] Liu S, Wang S, Zhu F, et al. HYDRA: large-scale social identity linkage via heterogeneous behavior modeling[C]// ACM SIGMOD International Conference on Management of Data. ACM, 2014:51-62.

[6] Ponzetto S P, Strube M. WikiTaxonomy: A Large Scale Knowledge Resource[J]. 2008:751-752.

[7] Wu F, Weld D S. Automatically Refining the Wikipedia Infobox Ontology[C]// WWW. 2008:635-644.

[8] Hoffart J, Suchanek F M, Berberich K, et al. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia[J]. Artificial Intelligence, 2013, 194:28-61.

[9] TW. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: A probabilistic taxonomy for text understanding,"[C] in SIGMOD 2012, pp. 481–492,ACM, 2012

[10] Zhou M, Bao S, Wu X, et al. An Unsupervised Model for Exploring Hierarchical Semantics from Social Annotations[C]// International the Semantic Web and, Asian Conference on Asian Semantic Web Conference. Springer-Verlag, 2007:680-693.

[11] Tang J, Leung H F, Luo Q, et al. Towards ontology learning from folksonomies[C]// International Jont Conference on Artifical Intelligence. Morgan Kaufmann Publishers Inc. 2009:2089-2094.

[12] Si X, Liu Z, Sun M. Explore the Structure of Social Tags by Subsumption Relations[C]// The. 2010:1011-1019.

[13] Lin H, Davis J, Zhou Y. An Integrated Approach to Extracting Ontological Structures from Folksonomies[C]// European Semantic Web Conference on the Semantic Web: Research and Applications. Springer-Verlag, 2009:654-668.

[14] Wang H, Wu T, Qi G, et al. On Publishing Chinese Linked Open Schema[M]// The Semantic Web – ISWC 2014. Springer International Publishing, 2014:293-308.

[15] Zhu J, Shen B, Cai X, et al. Building a Large-scale Software Programming Taxonomy from Stackoverflow[C]// The, International Conference on Software Engineering and Knowledge Engineering. 2015:391-396.

[16] Zhang C, Jiang S, Chen Y, et al. Fast Inbound Top-K Query for Random Walk with Restart[C]// Joint European Conference on Machine Learning & Knowledge Discovery in Databases. Mach Learn Knowl Discov Databases, 2015:608.

[17] Eto M. Document retrieval method using random walk with restart on weighted co‐citation network[J]. Proceedings of the Association for Information Science & Technology, 2015, 51(1):1-4.

[18] Yu W, Mccann J. Random Walk with Restart over Dynamic Graphs[C]// IEEE, International Conference on Data Mining. IEEE, 2017:589-598.