World Wide Web
https://doi.org/10.1007/s11280-018-0526-9

SCSMiner: mining social coding sites for software developer recommendation with relevance propagation

Yao Wan[1,2] · Liang Chen[3] · Guandong Xu[4] · Zhou Zhao[1] · Jie Tang[5] · Jian Wu[1,2]

Received: 1 July 2017 / Revised: 26 November 2017 / Accepted: 7 January 2018

Abstract With the advent of social coding sites, software development has entered a new era of collaborative work. Social coding sites (e.g., GitHub) can integrate social networking and distributed version control in a unified platform to facilitate collaborative developments over the world. One unique characteristic of such sites is that the past development experiences of developers provided on the sites convey the implicit metrics of developer's programming capability and expertise, which can be applied in many areas, such as software developer recruitment for IT corporations. Motivated by this intuition, we aim to develop a framework to effectively locate the developers with right coding skills. To achieve this goal, we devise a generativ e probabilistic expert ranking model upon which a consistency among projects is incorporated as graph regularization to enhance the expert ranking and a perspective of relevance propagation illustration is introduced. For evaluation, StackOverflow is leveraged to complement the ground truth of expert. Finally, a prototype system, SCSMiner, which provides expert search service based on a real-world dataset crawled from GitHub is implemented and demonstrated.

Keywords SCSMiner · Social coding sites · Expert finding · Developer recommendation · Relevance propagation

This article belongs to the Topical Collection: *Special Issue on Deep Mining Big Social Data*
Guest Editors: Xiaofeng Zhu, Gerard Sanroma, Jilian Zhang, and Brent C. Munsell

✉ Guandong Xu
guandong.xu@uts.edu.au

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou, China

[2] Realdoctor Artificial Intelligence Research Center, Zhejiang University, Hangzhou, China

[3] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China

[4] Advanced Analytics Institute, University of Technology, Sydney, Australia

[5] Department of Computer Science and Technology, Tsinghua University, Beijing, China

Published online: 05 February 2018

1 Introduction

Along with the prevalence of social networks in the world, social coding sites (SCS) is changing software development toward a more collaborative manner by the way of integrating social media functionality and distributed version control tools. For instance, as a large website for social coding, GitHub[1] is developing fast and growing into one of the best-known websites. Ever since the publication of Git and its successful application by Ruby community for distributed version control, the size of GitHub has been increasing since 2008 and this trend has dramatically accelerated over the past few years. According to our statistical analysis, until June 2015, the numbers of developers and projects in GitHub have reached up to 11,610,094 and 20,598,603, respectively.

A systemic view of social coding sites Figure 1 describes the schema of developer and project profiles in SCS. In SCS, information concerning user's name, company, location and followers count property is available, and the name, owner, description property and README file of project are also provided. Specifically, for developers, the *location* feature records where the developer is; the *hireable* feature records the current working status of a developer. For projects, the *contributors* feature records a project's contributors and lines of codes each contributor contributes to; the *language* feature (e.g., C++, Java, Python) refers to the main language of a project; the *stargazers_count* is the number of developers who have starred the project, and it can represent the quality of projects in some sense. It's worth noting that every project in SCS consists of a brief description, apart from some detailed information in the README file for most projects. Unlike traditional open source code platforms such as SourceForge[2] and BitBucket,[3] SCS like GitHub offers not only a code hosting service, but also an online tool for collaborative software development. In SCS, collaborators of projects can make changes to the content of repository and review the contributions submitted to the repository, while those who are not collaborators but wish to contribute to a project can *fork* it. In addition, users in GitHub can *follow* others in order to be notified of their actions, and users can *star* interesting repositories if they want to bookmark for later reference. Based on the information in SCS, we can construct a heterogeneous collaborative network of developers and projects.

Motivation The past development experiences of developers in SCS demonstrate the implicit metrics of developer's programming capability and expertise which can be applied in many areas such as software developer recruitment for IT corporations. In IT industry, IT professional recruiting has always been a slog for everyone involved, so finding better and easier ways to achieve successful recruiting is of great importance. The most common practice in current recruitment is through LinkedIn service by referring to the information published in LinkedIn to measure the professional level of targeted candidates. One limitation of such practice is that a personal profile can only be updated intentionally, i.e., it's a static profile rather than a dynamic profile. Another limitation is that it's difficult to measure the proficiency of developers only through static profile, e.g., recruiters can't judge the degree of proficiency of developer who declares that he/she is proficient in Java programming. In contrast, the expertise information derived from SCS will undoubtedly be

[1] https://www.github.com

[2] http://www.sourceforge.net/

[3] http://www.bitbucket.org

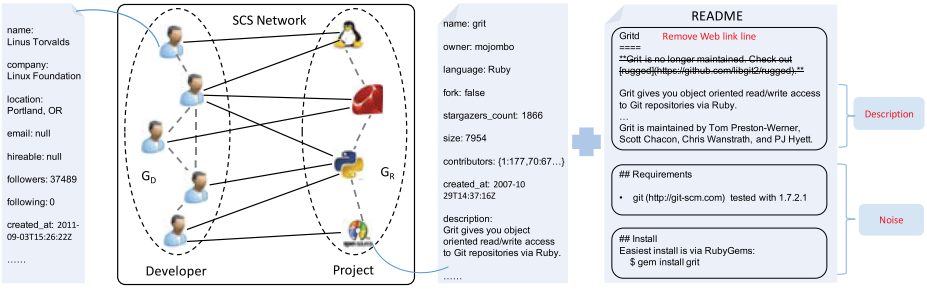


Figure 1 The schema of developer and project profiles in GitHub, one of the most popular social coding site. In GitHub, each user has name, company, location and followers count property, each project has name, owner, description and README property. A heterogeneous network between users and projects can be constructed based on these information

more dynamic and accurate about developers' portfolio, professional interest and influence through their participated projects hosted in the site, which makes recruiters easier to make selections. This intuition mainly motivate us to conduct this study for developing a practical system to fulfill such aim.

Challenges Bearing the above in mind, we propose to develop a coding expert recommendation system via SCS mining, named SCSMiner.[4] This system is able to provide a list of developers with ranking based on the given query and the rich information embedded in SCSs, which we firmly believe will benefit the software developers online recruitment and promote the development of open source community [19]. Although many SCSs such as GitHub has its own search engine to rank developers based on some simple metrics, e.g., the number of their followers, its performance is far from satisfaction. Having a certain number of followers usually indicates the popularity and professional competence of users to some extent, however having less followers does not mean that the candidate is necessarily uncompetitive. This work aims to address this ranking problem for a given query. There are mainly two challenges existing in our work. On the one hand, the textual information in projects is chaotic and limited. In the README files of projects, there exists many codes and installation instructions, making the topic model which performs well in textual information not satisfactorily in this context. On the other hand, existing expert finding models heavily rely on the textual information of projects alone, ignoring the implicit information in the network structure of projects and developers. Combining the embedded network structure along with the limited textual information has proved to an option to alleviate the limited textual information problem of projects [31]. Therefore, the challenge is how to integrate the textual information with the network structure in a unified framework to enhance the performance of expert searching and ranking simultaneously. Another difficulty is that unlike the academic network (e.g., DBLP) that many researches have been conducted on, the expertise of retrieved developers is difficult to quantitatively evaluate for the reason that no ground truth is available.

To implement this system, we design an expertise ranking algorithm. We first extract the textual information of projects from README file and then use a vector space model with cosine similarity to calculate the semantic similarity between a project and a given

[4] http://scsminer.wanyao.me

query. Furthermore, a graph based regularization framework is proposed to incorporate consistency hypothesis among the network structure of projects alongside probabilistic model to measure the expertise score of each candidate developer. Comprehensive experiments conducted on a real-world dataset demonstrate the performance of our proposed approach. Finally, a prototype system of SCSMiner is developed and demonstrated.

Contributions The main contributions of this paper can be summarized as follows.

- To the best of our knowledge, this work is the first attempt to discover local experts on social coding sites, which can benefit many applications.
- We apply a probabilistic model to evaluate the expertise of SCS developers and extend this model by graph regularization to incorporate the consistency hypothesis among projects. Furthermore, we also illustrate the inner connections between those two models from a relevance propagation perspective.
- Comprehensive experiments based on real-world data crawled from GitHub are conducted to demonstrate the performance of the proposed approach. Due to the lack of ground truth, we use the experts from StackOverflow to verify the experts retrieved by our model creatively.
- Our prototype SCSMiner providing expert search service is built based on the proposed approach. The involved data, codes have been released, which will facilitate other researchers to repeat our experiments and verify their own idea.[5]

Organization The remainder of this paper is organized as follows. Section 2 highlights some works related to this paper. Section 3 defines our problem mathematically and gives an overview of SCSMiner's system architecture. In Section 4, a probabilistic model is proposed to measure the expertise of each candidate related to a given query. Consistency hypothesis are proposed and incorporated to improve the performance of expertise ranking in Section 5. Section 6 describes the dataset we use in our experiment and shows the experimental results and analysis. Finally, we conclude this paper and propose some future research directions in Section 7.

2 Related work

In this section, we will make a thorough investigation of existing works about the social coding sites and present some related works about it. Generally, the related works can be grouped into three categories.

Mining on social coding sites The important role of social network played in software development and the transparency and collaboration the social network brings have been confirmed in [4, 8]. Social coding sites have been studied from several different perspectives. The first is the study of social network among social coding sites [15, 16, 25], where the collaboration between users, the dissemination of projects and the interaction among them are analyzed systematically. In those works, the social coding sites are seen as a kind of social network. The second is on mining of source code and commit log. For instance, [12] aims to characterize the known coding errors to improve the automatic detection of

[5] http://wanyao.me/projects/scsminer.html

software defects, [27] quantitatively and qualitatively investigates when and why developer change software licenses. These works can be used to promote the development of software engineering. Besides, some interesting related applications are also emerging. In [13], the authors extract the professional skill of developers in GitHub and propose a pipeline that automatizes the process of recruiting and job recommendation. For software development, [18] investigate the team formation problem for better collaboration. In [19], the authors analyze the impression formation in an online distributed software development community with social media functionality. In [26], the authors investigated the interplay between StackOverflow activities and the development process, reflected by code changes committed to the largest social coding repository GitHub. Different from those works, our paper mainly focus on finding the experts by their domain knowledge and social network.

README extraction and text analysis In [3], a new statistical approach is introduced to automatically partitioning text into coherent segments. Sodedant et al. [22] develop an approach to learn text extraction rules automatically for text styles ranging from highly structured to free text. In [28], a novel measure is proposed for semantic parsing evaluation of interpreters for instructions in computer program README files. The description information extracted from README file are used to calculate the semantic similarities in our model. Some models can handle it. A non-probabilistic language model [20] based on TF-IDF is proposed. ln [14], Hofmann et al. propose the probabilistic latent semantic indexing (pLSI) and apply it to information retrieval. Blei et al. [5] put forward a generative probabilistic model called Latent Dirichlet Allocation (LDA). Based on these topic models, some other works have been conducted for modeling both author interests and document contents. The Author-Topic model [23] extends the LDA by integrating the authorship and can find a topic mixture over documents and authors. Jie et al. [24] proposed a unified topic model to simultaneously model the topical aspects of different types of information in the academic network. Conducting experiments using those classical methods, we find that the simplest TF-IDF method achieves the best performance.

Expertise finding Broadly related to our scenario is expert finding. Considerable works have been conducted to associate query topics to people for expertise retrieval [2]. In [10], the authors propose a general probabilistic framework for studying expert finding problem and derived two families of generative models e.g., candidate generation models and topic generation models from the framework. In [11], a discriminative learning framework is proposed for expert finding and two specific probabilistic models i.e., the arithmetic mean discriminative model and the geometric mean discriminative model are derived from this framework. In [17], the authors propose a novel voting model for predicting and ranking candidate expertise with respect to a query inspired by techniques from the field of data fusion. In [21] the authors model the multi-step relevance probability dissemination in topic-specific expertise graphs consisting of persons and top retrieved documents. In [9], the authors propose a joint regularization framework to enhance expertise retrieval by modeling heterogeneous networks as regularization constraints on top of document-centric model. With the succesfully development of deep learning, some deep learning based method such as RNN is utilized to learn the representation of users' topics [30]. In [30], Zhao et al. propose a ranking metric network learning framework for expert finding in community question answering by exploiting both users' relative quality rank to given questions and their social relations. Our work mainly refers to [9] but not limited to. Although our proposed model is mainly borrowed from [9], we transfer it to a novel scenario that is expert finding in social

coding sites and analyze the topics of developers from the README files of their developed projects. Besides, we discuss the connection between our proposed model and generative probabilistic model from the perspective of random walk. Furthermore, we design a novel method to indirectly evaluate the performance of proposed model.

3 Preliminaries

In this section, we introduce the information we can get from a SCS, and formally define the problem of expertise search and ranking in a heterogeneous network.

3.1 Problem definition

As shown in Figure 1, a collaborative heterogeneous network consists of two types of object sets, including a developer set $D = \{d_1, d_2, \cdots, d_m\}$ and a project set $R = \{r_1, r_2, \cdots, r_n\}$. Such a heterogeneous network can be defined as $G =< V, E >$, where $V = V_D \bigcup V_R$ and $E = E_D \bigcup E_R \bigcup E_{D,R}$. This network can be decomposed into three subnetworks, G_D is a collaborative network between developers, G_R is a network of projects, while $G_{D,R}$ is a network representing the relationship between developers and projects.

In a developer-developer network G_D, each node represents a given developer. A link is assigned to two nodes when the corresponding developers collaboratively develop at least one common project. We associate a weight to each edge taking into account the number of projects where the two developers work together. For a project-project network G_R, each node represents a given project, two nodes are connected if the corresponding projects have at least one common developer. In developer-developer network, we associate a weight to each edge taking into account the number of common developers the projects have. For $G_{D,R}$, a link is assigned, if there exists a "*develop*" or "*developed by*" relationship between a developer and a project.

Given a heterogeneous network between developers and projects, as well as the profile information of each project and developer, and the collaborative relationship between them, the goal is to learn the expertise vector $f \in R^D$, where each element corresponds to the expertise of each developer given a query.

3.2 An overview of system architecture

Based on the model proposed in this paper, we implement an expert search system on social coding sites named SCSMiner. The initial version of this system is accessible on http://scsminer.wanyao.me. Figure 2 shows the system architecture of our SCSMiner search system. The system consists of four main components.

- **Extraction:** The crawler(s) crawls all the user and project information from social coding sites, and the features which will be used in our model are extracted. Due to the README file of projects information is chaotic with many codes and installation instruction, some special processes are conducted to extract descriptions from README files.
- **Storage and Access:** This component is the data center of SCSMiner. The information of users, projects and the relationship between them are recorded here, and this component provides an interface for invoking. Specifically, we employ Mongodb for storage and inverted file indexing method for index.

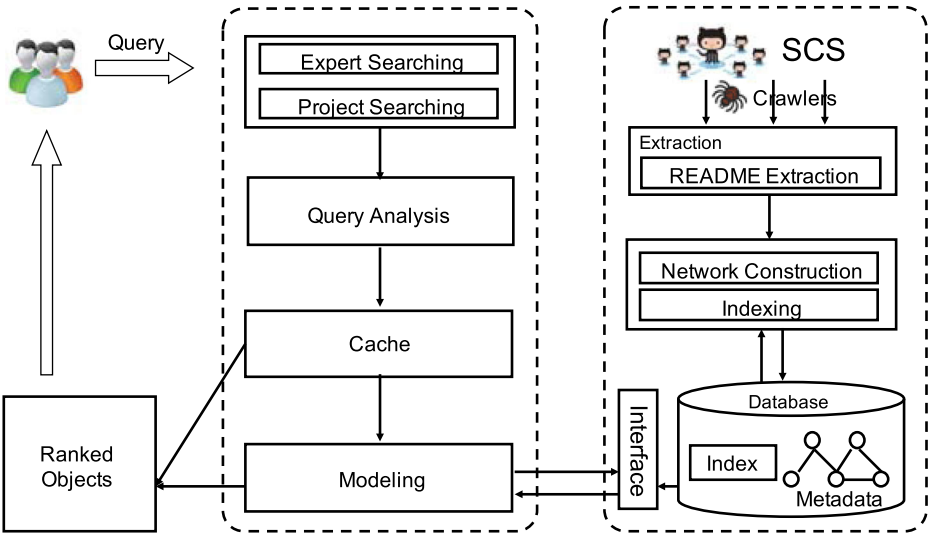


Figure 2 The system architecture of SCSMiner

- **Modeling:** It utilizes a probabilistic model to measure the expertise of each developer for a given query basically, and a regularization framework is utilized to incorporate one consistency existing in the heterogeneous network between developers and projects.
- **Search Services:** Based on the modeling results, SCSMiner provides its search service to client users.

4 Generative probabilistic model for expertise ranking

Generative probabilistic models are efficient approaches in expertise retrieval as their good empirical performance and their potential for incorporating various extensions in a transparent and theoretically sound fashion. The probabilistic model proposed in this section follows the basic idea of a document-centric probabilistic model, which is proposed to estimate the expertise of a candidate by summing the relevance of its associated documents [1, 9]. In this model for a given query q, the relevance probability of candidate developer d is determined by the following equation:

$$p(d|q) = \frac{p(q|d)p(d)}{p(q)}, \tag{1}$$

in which, $p(d|q)$ denotes the expertise score of candidate developer d for the given query q; $p(q|d)$ is the probability of query q given the candidate developer d; $p(d)$ is the prior probability of being expert for an expert candidate d, and $p(q)$ is the prior probability of a given query q. As $p(q)$ is a constant in expert ranking, it can be ignored from above equation. Therefore, relevance probability of each experts can be estimated by the probability of a query given expert candidate, weighted by the prior probability that expert candidate d is an expert:

$$p(d|q) \propto p(q|d)p(d). \tag{2}$$

According to the document centric model [1], the expertise score of a developer related to a given query can be formulated as:

$$p(q|d) = \sum_{r \in \mathscr{R}_d} p(r|d)\, p(q|r, d), \tag{3}$$

where $\mathscr{R}_d$ indicates the subset of projects associated with the candidate developer d. In this equation, to simplify the calculations, it is assumed that candidate developer d is conditionally independent of the query q given project r (i.e., $p(q|r, d) \approx p(q|r)$). As a result, by substituting $p(q|d)$ in (2) and expanding $p(r|d)$, $p(d|q)$ can be estimated as follows:

$$\begin{aligned} p(d|q) &= \sum_{r \in \mathscr{R}_d} p(q|r)\, p(r|d)\, p(d) \\ &= \sum_{r \in \mathscr{R}_d} p(q|r) \frac{p(d|r)\, p(r)}{p(d)}\, p(d) \\ &= \sum_{r \in \mathscr{R}_d} p(q|r)\, p(d|r)\, p(r), \end{aligned} \tag{4}$$

where $p(r)$ denotes the prior relevance probability of project r, $p(q|r)$ is the probability of a query q given the project r, and $p(d|r)$ is the probability of the association between the project and the candidate developer d. The probability $p(d|r)$ provides a ranking of candidates associated with a given project r, based on their contribution made to project r. According to (4), in order to rank candidate developers, we should estimate three probabilities, namely, prior probability of retrieval for project r (i.e., $p(r)$), relevance probability of project r (i.e., $p(q|r)$), and association probability of project r and candidate developer d (i.e., $p(d|r)$).

In (4), $p(q|r)$ denotes the semantic similarities between project r and a given query q. In our paper, we simply apply vector space represented by TF-IDF determine $p(q|r)$, where TF-IDF is a typical vector space model and it is defined as:

$$TF\text{-}IDF(w, r, R) = f(w, r) \times log \frac{|D|}{|\{r \in R : w \in r\}|}, \tag{5}$$

where $f(w, r)$ is the frequency of word w in project r, $|R|$ is the total number of documents in the corpus, $|\{r \in R : w \in r\}|$ is the number of documents in which the term w appears.

In our paper, we use cosine similarity to determine the semantic similarity between a given query and project. The cosine similarity is defined as follows:

$$sim(\mathbf{q}, \mathbf{r}_j) = \frac{\mathbf{r}_j \cdot \mathbf{q}}{\|\mathbf{r}_j\| \cdot \|\mathbf{q}\|} = \frac{\sum_{i=1}^{n} w_{ij} w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} \sqrt{\sum_{i=1}^{n} w_{iq}^2}}, \tag{6}$$

where $\mathbf{q}$, $\mathbf{r}_j$ are the word vector of query q and project r_j respectively.

For a project r developed by multiple developers, we assume that each developer has the same level of knowledge about the topics described in the project and therefore the association probability of project r and candidate developer d is estimated as follows:

$$p(d|r) = \begin{cases} \frac{c_d}{\sum_i^D c_i} & d \textit{ is a contributor of } r, \\ 0 & \textit{otherwise}. \end{cases} \tag{7}$$

where c_d is the number of commits submitted by developer d, $\sum_i^D c_i$ is the total number of commits submitted to project r.

In addition, $p(r)$ can be seen as the quality of project. Indeed, $p(r)$ can be set to be the combination of star number and total lines of code of a project, which can be represented as followers.

$$p(r) = \eta s_r + (1 - \eta) l_r, \tag{8}$$

where s_r and l_r is the stars count and total lines of code of the project r, respectively; η is the tuning parameter that controls the weight between quality coming from the stars count and lines of code. In our experiments, we scale the s_r and l_r to a same range via min-max noamalization, and η is set to be 0.5.

For simplicity, let $\mathbf{x}$ be the relevance vector between project r_i and query q with $x_i = p(q|r_i)$, $\mathbf{Q}_R$ be the diagonal matrix that represents the project quality, and $\mathbf{P}_{RD}$ be the composition matrix between projects and developers. Then the primary model as shown in (4) can be rewritten as:

$$\mathbf{f} = \mathbf{P}_{RD}^T \mathbf{Q}_R \mathbf{x} \tag{9}$$

where $\mathbf{f}$ represents the relevance score vector of all candidate developers. The underlying intuition of this model is to estimate the expertise of candidate developers based on the relevance and quality of associated projects.

5 Modeling the network

In the previous section we propose a probabilistic model to determine the expertise score of each developer. As shown in Figure 3a, the generative probabilistic model can be considered as an one-step relevance propagation [21] from projects to related candidate developers. In this model, only textual information of projects are used to calculate the relevance between query and projects, ignoring the network structure among projects. In this section, we propose a consistency among projects and extend the probabilistic model to incorporate project consistency via graph regularization. The intuition is that relevance can not only propagate from projects to developers, it can also propagate among projects, which is considered as a consistency in our paper. Figure 3 gives a graphical illustration of connections between generative probabilistic model (PM) and that with graph regularization (regPM).

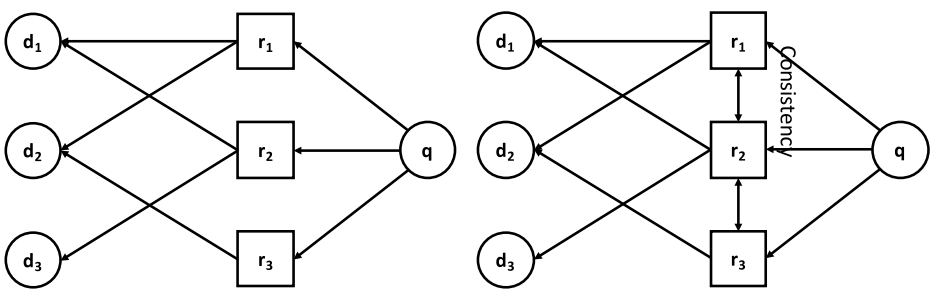


Figure 3 Graphical illustration of connections between generative probabilistic model and that with relevance propagation

5.1 Project consistency regularization

According to our common sense and observation, we propose the following consistency hypothesis.

Project consistency hypothesis *Usually similar projects tend to be of similar relevance with respect to a given query*. In the project layer network, it is reasonable to assume that the neighbors of project r are those projects that are similar to it.

In the generative probabilistic model, we need to calculate the semantic similarity between query and project. However, as declared before, the textual information of projects are limited. Incorporating the project consistency will connect the neighbor projects, and the similarity of similar projects will consistent with each other, this will alleviate the limited textual information problem. In this subsection, we will enforce the consistency hypothesis proposed above with the probabilistic model by defining the regularization constraints.

Graph regularization is an effective technique in the semi-supervised learning where the goal is to estimate the labels of unlabeled data using other partially labeled data and their similarities. For project consistency hypothesis in our scenario, the goal is to refine relevance score vector $\mathbf{x}$ based on the project consistency regularization that *similar projects tend to be of similar relevance with respect to a given query*. In the probabilistic model, the initial relevance score $\mathbf{x}^0$ can be determined by the vector space model. According to [31], the problem can be addressed by minimizing the regularization loss below:

$$\Omega(\mathbf{x}) = \mathbf{x}^T(\mathbf{I} - \mathbf{S_R})\mathbf{x} + \mu_R \left\| \mathbf{x} - \mathbf{x}^0 \right\|^2 \tag{10}$$

where $\mu_R > 0$ is the regularization parameter, $S_R \in \mathbb{R}^{|R\times R|}$ denotes the pairwise similarities among projects G_R. The first term of the loss function defines the *project consistency*, which prefers small difference in relevance scores between nearby projects; the second term is the *fitting constraint*, which measures the difference between final relevance scores $\mathbf{x}$ and the initial relevance scores $\mathbf{x}^0$. The initial relevance score vector $\mathbf{x}^0$ can be calculated according to (9) in the probabilistic model.

Minimizing $\Omega(\mathbf{x})$ will force the neighbor projects to have similar relevance scores. Setting $\partial\Omega(\mathbf{x})/\partial\mathbf{x} = 0$, we can see that the solution $\mathbf{x}^*$ is essentially the solution to the linear equation:

$$(\mathbf{I} - \alpha\mathbf{S}_R)\mathbf{x}^* = (1-\alpha)\mathbf{x}^0 \tag{11}$$

where $\alpha = 1/(1+\mu_R)$. In this equation, we need to calculate the inverse matrix $(I-\alpha\mathbf{S}_R)^{-1}$. Fortunately, matrix $\mathbf{S}_R$ is always very sparse, causing calculation to be costly. One iterative solution to this equation is given in a related work using a power method [32].

$$\mathbf{x}(t+1) = \alpha\mathbf{S}_R\mathbf{x}(t) + (1-\alpha)\mathbf{x}^0 \tag{12}$$

where $\mathbf{x}^* = \mathbf{x}(\infty)$ is the solution. Here $\mathscr{L} = (\mathbf{I} - \alpha\mathbf{S_R})$ is essentially a variant Laplacian on this graph using S_R as the adjacency matrix; and $\mathscr{K} = (\mathbf{I} - \alpha\mathbf{S_R})^{-1} = \mathscr{L}^{-1}$ is the graph diffusion kernel.

Now the interesting question is how to calculate $\mathbf{S}_R$ in R. For graph data, several works [7] borrow results form spectral graph theory for obtaining the similarity measures. Specifically for an undirected graph, $\mathbf{S}_R$ is simply the normalized adjacency matrix $\mathbf{W}$:

$$\mathbf{S}_R = \Pi_R^{-1/2}\mathbf{W}_R\Pi_R^{-1/2} \tag{13}$$

where $\mathbf{W}$ is the adjacency matrix of projects in G_R, $\mathbf{W}_{ij} = 1$ if node i is linked to node j, otherwise, $\mathbf{W}_{ij} = 0$, and Π is a diagonal matrix with $\Pi_{ii} = \sum_j \mathbf{W}_{ij}$.

5.2 Connections and discussions

In the regularization framework, suppose $\alpha = 0(\mu_R \to \infty)$; as shown in (10), Ω puts all weight on the second term $\left\|\mathbf{x} - \mathbf{x}^0\right\|^2$ to ensure that the final scores are equal to the initial scores $\mathbf{x}^0$. In this case, the regularization framework boils down to the baseline generative probabilistic model.

The regularization framework can also be interpreted as *lazy random walks* with the transition probability matrix $\mathbf{P} = (1-\rho)\mathbf{I} + \rho\mathbf{S}_R^{-1}\mathbf{W}_R$ where ρ is a parameter in (0, 1) [33]. This means, with probability ρ, following on a link coincident with the vertex of the current position and chosen with probability proportional to the weight of the link, and with the probability $1-\rho$, it stays at the current position. This can also be observed from Figure 3. Comparing with the generative probabilistic model, the regularized framework introduce more random walk rules (e.g., walks among projects).

6 Experiments

6.1 Dataset collection

As we describe before, our experiments are conducted on the GitHub dataset. However, the most significantly difficulty is the lack of ground truth of whether the retrieved developer is expert or not. One simple approach is to select some queries and judge the relevance of retrieved results manually. However, this approach can not be implemented in large scale and may be affected by many human factors. As such, we deliberately utilize StackOverflow[6] as an indirect ground truth to conduct evaluation.

6.1.1 Dataset1: intersection of github and stackoverflow

According to [26], expertise derived from StackOverflow can somehow reflect those from GitHub indirectly. In [26], the authors investigate the interplay between StackOverflow activities and the development process reflected by code changes committed to GitHub. They find that the StackOverflow activity rate correlates with the code changing activity in GitHub. So we extend their finding to validate our model via StackOverflow. Here the key is first to identify the developers both appeared in GitHub and StackOverflow, then to verify the developers recommended by our GitHub-base system against the results given by StackOverflow.

GitHub GitHub, one of the most popular social coding sites, has gained much popularity among a large number of software developers around the world. It has a publicly accessible API. Crawling GitHub website by its API, we get 28,362,019 projects, 15,647,255 users and make out the relationships between them.

[6] http://stackoverflow.com/

Table 1 Top 20 tags in StackOverflow

Tag	Frequency	Tag	Frequency
javascript	182,509	ruby	55,463
php	146,926	ios	51,025
java	144,660	css	49,756
c#	143,001	mysql	49,642
python	120,203	.net	48,523
jquery	118,475	objective-c	45,180
c++	79,503	iphone	43,805
android	76,592	c	39,789
ruby-on-rails	74,207	asp.net	36,026
html	70,047	sql	32,371

StackOverflow StackOverflow is a popular online programming question and answer community started in 2008. It dumps and releases the dataset every three months. The data used in our experiment is released in August 2012, containing about 1,295,622 registered users dating from July 2008 to August 2012.

Intersection To intersect data from GitHub and StackOverflow, a conservative approach matching email address is adopted in our experiment. In the GitHub dataset email address are present, while in the StackOverflow dataset email addresses are obscured, but their MD5 hashed are available. Therefore, we merge a GitHub and a StackOverflow user if their MD5 email hashes are identical. Furthermore, for computation simplicity, only those users whose reputation in StackOverflow is greater than 5 and followers number in GitHub is greater than 10 are considered. Finally, we obtain 16,567 users and 458,639 related projects.

Ground truth In StackOverflow, each user answers many programming problems and the tags and their corresponding count of those problems they answer are collected.[7] In terms of the retrieved results, we consider the retrieved developer who has also answered some corresponding problems in StackOverflow as relevant, and the count of answering questions can be taken as the degree of relevance. Actually, we divide the degree of relevance into five degrees (0–4) by four thresholds (e.g., 0, 5, 10, 50). In this evaluation, top 50 queries in StackOverflow are used as queries (the top 20 queries are shown in Table 1).

6.1.2 Dataset2: human labeled data

Since the some popular users in GitHub may not appear in StackOverflow, and to have a clear understanding on the retrieved results, we also evaluate our model on some popular users. We extract projects whose star number is greater than 400 firstly, and then extract users who make contributions to the projects. Finally, we obtain 11,437 projects and 158,646 users.

Ground truth Evaluation on this dataset is also a nontrivial task since the ground truth is difficult to obtain. Here, we consider this strategy. We first random select 8 popular queries (e.g, *linux*, *javascript*, *app*, *plugin*, *angular*, *framework*, *game*, *image*) a developer may be interested in. Then a pooling methods (see Section 6.3.2) are used to select an initial set of

[7] https://api.stackexchange.com/2.2/users/1335234/tags?order=desc&sort=popular&site=stackoverflow

developers for judgment. Then we invited 4 computer science graduated students to judge our retrieved result. The followers count and the projects the developer contributed to are two criteria the judge referred when determining the relevance of developers to queries. As it's difficult to measure the degree of relevance by human, the judges just need to determine whether the candidate developer is expert or not, so the NDCG metric is not calculated on this dataset. The queries and their corresponding expert used in our experiment are listed in http://wanyao.me/projects/scsminer.html.

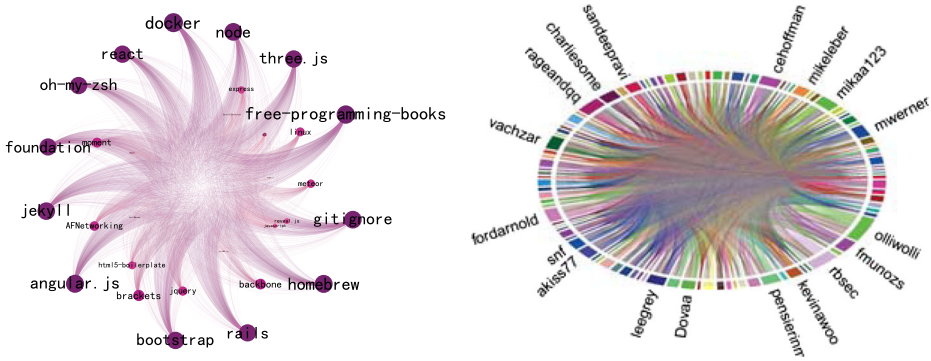
6.1.3 Dataset characteristics

Figure 4 presents an overview of the network structure of GitHub and the collaboration relationship between developers. Figure 4a shows the number of participants in each project (the size of each node is related to its indegree). From this figure, we observe that the number of participants in projects such as "*foundation*", "*node*", "*three.js*" is maximal. To further study the relationship between the number of participants and projects, we examine each project's popularity by its stars, from which we notice that for most projects, the number of participants indicates its popularity.

Figure 4b illustrates relationships between developers where each sector represents a developer. In this figure, we observe that the sectors indicate the developer's activity degree of coorperation with others. We note that *olliwolli* and *vachzar* are the most active user in out GitHub dataset.

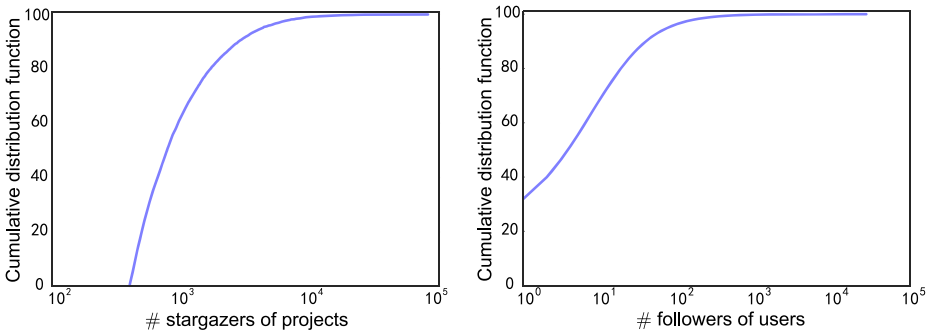
Figure 5 shows the cumulative distribution function of star count with respect to projects as well as the cumulative distribution function of followers number with respect to developers. Figure 5a shows that in our experimental dataset, most projects have a little more than 400 stars, only few developers have many stars. From Figure 5b, we note that the distribution of followers roughly follows a power law. This indicates that only few developers have more 10 followers; most developers have few or none.

Table 1 lists the Top 20 tags in StackOverflow. The frequency column in this table represents the number of question which are tagged with the corresponding tag. From this table, we can find that most tags in Q&A sites are programming languages like *java*, *php* and *java*, which is in accordance with our expectation.



(a) Network overview **(b)** Collaboration network

Figure 4 The GitHub network. **a** An overview of network structure in GitHub. **b** Collaboration network between developers in GitHub



(a) CDF of # stars of projects **(b)** CDF of # followers of users

Figure 5 **a** The cumulative distribution function of projects with respect to stars number. **b** The cumulative distribution function of developers with respect to followers number

6.2 Preprocessing

Through statistics of 1,406,409 README files, the README files fall into three main three categories: *markdown* (89.14%), *txt* (7.4%) and *html* (0.008%). Since the number of *html* README files is limited, we consider them as *txt* files simply. Regarding the other two types, the process consists of 3 steps: (1) separating the README file into paragraphs; (2) selecting the description paragraphs; and (3) removing unnecessary parts.

In the paragraph separation step, for *markdown* files, by utilizing regular expressions, we separate the whole README file by the subtitle separator in *markdown* (e.g., '##', '==', '–') into two sequences. One is the separator sequence which only contains separators, the other is the words sequence. Then we check the separator type. If it is "#", combining the "#" with words to obtain the paragraph from the two sequences. If it is "-" or "=", things may be complex. The subtitle of the paragraph has been split into the previous paragraph, which is its last line. So we have to choose the last line of the last paragraph as the new subtitle, then combine it with the separator ('–' or '==') and the sequence into a new paragraph. For *txt* files, it's hard to get a common separator. The method for finding a subtitle is to judge each line of text by four conditions: (a) Delete the lines contain a Web link. It is probably the download link won't be useful for description. (b) The whole line is English characters from *A-Z* or *a-z*. (c) The line has no special signals (e.g. ',' '.' '$') that seldom appear in subtitles (d) Shorter than 40 characters. If the line meets conditions (b), (c) or (d), we regard it as a subtitle. Then we can use it to separate the text.

When selecting description paragraphs, we only consider the first three paragraphs in a README file, because description text is usually fount there. We check and select the descriptions by using regular expression to search for key words in the subtitle. The key words we use are '*description*', '*feature*', the project name and other words that could represent the meaning of '*description*' (e.g. introduction). As for removing unnecessary parts, we remove Web links and common Linux commands (e.g., '*mkdir*', '*apt*'), which won't contribute to a useful description. After these three steps, the textual information can be extracted from the README file (Figure 1 gives a graphical illustration of these processes). According to our manual analysis based on 100 samples, the accuracy of the proposed approach reaches as high as 82.0%.

6.3 Experimental setup

6.3.1 Evaluation metrics

For our evaluation, several categories of Web search evaluation metrics are used to measure the performance of our proposed model from different aspects, including some relevance based metrics and ranking based metrics. To measure the relevance of our search results, we use the precision at rank k ($P@k$), which is widely used and is defined as: $P@k = \frac{\#\ relevant\ in\ top\ k\ results}{k}$. $P@k$ measures the fraction of the top-K retrieved results that are relevant to the given query. Another metric MAP is the mean value of the average precision computed for all queries. $MAP = \frac{\sum_{q=1}^{Q} AP(q)}{|Q|}$ and $AP = \frac{\sum_{n=1}^{N}(P@n \times rel(n))}{R}$, where n is the rank, N the number retrieved, and $rel(n)$ is a binary function indicating the relevance of a given rank. We use Mean Reciprocal Rank (MRR) to evaluate the ranking of our search results. A larger MRR value means a better result. The MRR is defined as $MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$, where $|Q|$ is the size of query set. Another ranking metric we use in our evaluation is normalized discounted cumulative gain (NDCG), which measures the performance of a retrieval system based on the graded relevance of the retrieved entities. The NDCG@k is defined as: $NDCG@k = \frac{1}{|Q|}\sum_{q=1}^{|Q|} Z_{kq} \sum_{j=1}^{k}\frac{2^{r(j)}-1}{log(1+j)}$, where Z_{kq} is a normalization factor calculated to make it so that a perfect ranking's $NDCG$ at k for query q is 1; $r(j)$ is the relevance score assessors give to retrieved entity for query q. Beside the measurement of precisions, bpref [6] is a good function that evaluates how frequently relevant documents are retrieved before non-relevant documents. It is defined as $bpref = \frac{1}{R}\sum_r\left(1 - \frac{|\text{n ranked higher than r}|}{R}\right)$, where R is the number of relevant documents, r is the relevant document retrieved and n is member of the first R non-relevant documents retrieved.

6.3.2 Comparison models

In this subsection, comparisons between the models described above and the state-of-the-art are made to show the effectiveness of our proposed approach.

- **Voting Model (Voting).** The voting model ranks candidates by the number of top projects they have contributed to [17]. In [17], the authors evaluate 12 voting techniques based on known data fusion techniques. Here, we only consider the reciprocal rank data fusion technique, which will highly rank candidates who contribute to many top projects.
- **Infinite Random Walk (IRW).** In [21], the authors propose three kinds of random walks based on the principle of multi-step relevance propagation in topic-specific expertise graphs. Here, only the infinite random walk is considered.
- **Probabilistic Model (PM).** This is the probabilistic model proposed in Section 4.
- **Probabilistic Model with Graph Regularization (regPM).** Based on PM model, this model incorporates the consistency hypothesis among projects with probabilistic model via graph regularization (see Section 5).

All the experiments in this paper are implemented with Python 2.7, and run on a computer with an 2.2 GHz Intel Core i7 CPU and 64 GB 1600 MHz DDR3 RAM, running Debian 7.0.

6.4 Experimental results

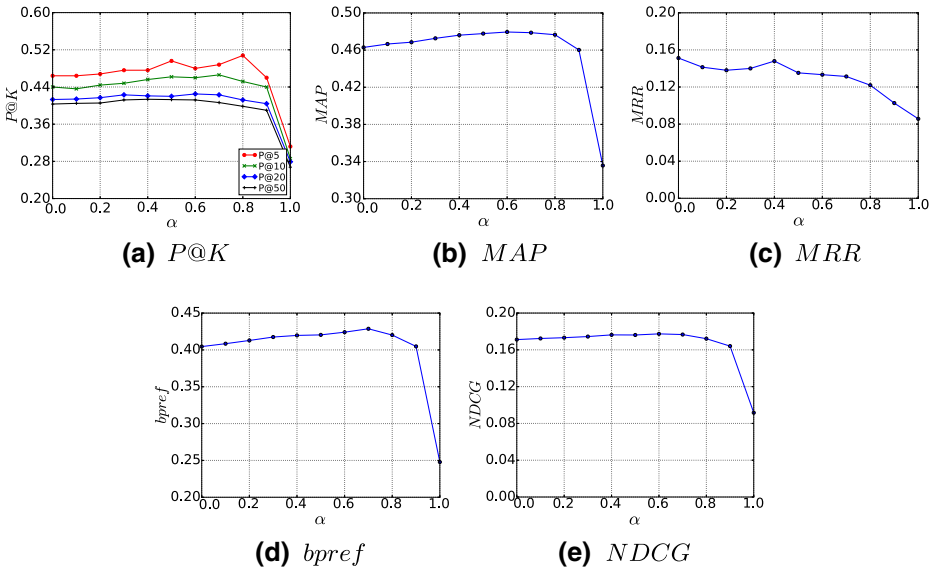
The experimental results of those comparison methods on two datasets are shown in Table 2. In this experiment, the regPM model achieves its best performance when we set $\alpha = 0.8$, $\#iteration = 10$ and $Top\text{-}K = 50$. Some conclusions can be drawn from this table. Firstly, the state-of-the-art voting model has a bad performance for the reason that it doesn't so well integrate the relevance between query and project as the proposed probabilistic model, nor does it integrate the quality of the the project and the contribution users make to the project. The voting model just takes the number of top projects they have contributed to into consideration. Secondly, when compared with voting model, the generative probabilistic model has obtained remarkable results by way of integrating the relevance between query and project, the quality of project and the contribution users make to the project. Thirdly, as described before, the PM model can be interpreted as one-step relevance propagation from projects to developers. IRW is a multi-step relevance propagation model which considers both the relevance propagation from developers to projects, and the relevance propagation from projects to developers. However, from the experimental results, we note that IRW model doesn't perform better than PM model in our dataset. This may be illustrated by that the propagation between projects and developers is unidirectional, thus the bidirectional propagation may reduce the performance instead.

Finally, comparing regPM with PM, we notice that integrating relevance propagation among similar projects contributes dramatically to performance improvement in many evaluation metrics, for example, the $P@5$ of regPM has reached up to 0.508 on Dataset1, which improves 9.48%, gains outstanding performance when compared with the PM model. The above results have verified our intuition that the network structure among projects is complementary and the consistency hypothesis characterizes the relevance propagation among projects well. Another interesting finding from the results is that on Dataset1 the MRR of regPM is smaller than that of PM model. We can illustrate this phenomenon from two aspects. On the one hand, it may be caused by the ground truth since the regPM performs better than PM in MRR on Dataset2. On the other hand, introducing the consistency among projects may dispersed the divergence of expert ranking, which may have an impact on the MRR metric.

Table 2 The experimental results of different models on different metrics

Methods / Metrics	P@5	P@10	P@20	P@50	MAP	MRR	bpref	NDCG
Dataset1								
Votes	0.100	0.160	0.173	0.158	0.210	0.051	0.104	0.034
IRW	0.448	0.428	0.405	0.394	0.447	0.145	0.395	0.170
PM	0.464	0.440	**0.413**	**0.403**	0.463	**0.151**	0.404	0.171
regPM	**0.508**	**0.452**	0.412	0.398	**0.476**	0.121	**0.420**	**0.172**
v.s. PM	+9.48%	+9.44%	−0.24%	−1.24%	+2.81%	−19.87%	+3.96%	+0.58%
Dataset2								
Votes	0.100	0.075	0.081	0.065	0.248	0.283	0.185	–
IRW	0.500	0.413	0.350	0.280	0.555	0.637	0.517	–
PM	0.775	0.738	0.650	0.515	0.710	0.200	0.690	–
regPM	**0.800**	**0.788**	**0.675**	**0.535**	**0.723**	**0.236**	**0.701**	–
v.s. PM	+0.65%	+6.78%	+3.85%	+3.88%	+1.83%	+18.00%	+1.59%	–

Best scores are in boldface



(a) $P@K$ **(b)** MAP **(c)** MRR

(d) $bpref$ **(e)** $NDCG$

Figure 6 Impact of α on different metrics

Impact of α In our model, $\alpha = 1/(\mu_R + 1)$ where μ_R is a regularization parameter that controls project consistency, i.e., similar projects tend to be of similar relevance with respect to a given query. To study the impact of α on the metrics of our approach, we change α from 0 to 1 with a step value of 0.1. We set $\lambda = 0.8$, $Top\text{-}K = 50$, and $\#iteration = 10$ in this experiment. Experiments are conducted on Dataset1. Figure 6 shows the impact α on different metrics.

From this figure, we can observe that the value of α has a significant impact on the performance of searching experts on SCSMiner and the optimal α lies between 0.4 and 0.8. As α increases, the metrics increase at first, but when α surpasses a certain threshold, they decrease with further increase in the value of α. This indicates that incorporating project consistency approximately may improve the performance of searching results. While when α approaches 1, the μ_R will approach 0, which indicates that the rule that the final relevance score $\mathbf{x}^*$ should be close to the initial score $\mathbf{x}^0$ is ignored. On this condition, the regPM model gets its worst performance.

6.5 Search example analysis

To gain a better insight into the proposed algorithm, we chose "linux" as an example query for showing detailed results. The top-20 developers are listed in Table 3. It is obvious that the results of the regPW model make more sense than those of the PW model. After looking into details, we see that, unlike their ranking in GitHub, and some other developer ranking in websites according to the "followers" number, although the followers number of a developer is small, it can still get a high rank in our method. This can be illustrated by the ranking mechanism in the probabilistic model and the consistency among projects we introduce in this paper. For example, *broonie* has a high rank, although his followers number is small. This is because he collaborated much with *torvalds* on the development of the *Linux* kernel source, with a contribution of 3%, which is as high as the contribution of *torvalds*.

Table 3 The top-20 experts with their corresponding number of followers retrieved by PM and regPM models

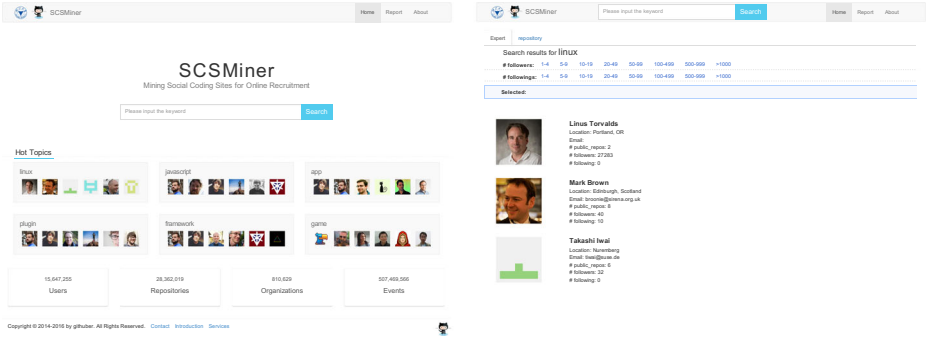
Query: linux							
Ground truth				PM		regPM	
Rank 1–10	#followers	Rank 11–20	#followers	Rank 1–10	Rank 11–20	Rank 1–10	Rank 11–20
torvalds	27,283	liewegas	97	**torvalds**	bzolnier	**torvalds**	arndb
broonie	40	lwfinger	87	**broonie**	**AxelLin**	**broonie**	ralfbaechle
tiwai	32	bigguiness	44	**tiwai**	arndb	**tiwai**	**jacknagel**
davem330	26	AxelLin	18	**davem330**	wishstudio	**davem330**	tj
gregkh	1,055	pmundt	18	**gregkh**	ralfbaechle	**gregkh**	olofj
0xAX	609	jingoo	17	jmberg	**jacknagel**	jmberg	**danvet**
BrewTestBot	23	kaber	12	**pmundt**	kovidgoyal	**pmundt**	**jingoo**
adamv	245	rjwysocki	10	**bigguiness**	olofj	**bigguiness**	**kaber**
rustyrussell	102	danvet	8	vladikoff	**danvet**	bzolnier	**0xAX**
jacknagel	108	tmlind	8	**0xAX**	**BrewTestBot**	**AxelLin**	larsclausen

Relevant experts are in bold-face

6.6 Demonstration

Based on the model proposed in this paper, we implement a search system based on GitHub named SCSMiner. This system is dedicate to mining of activities of developers on social coding sites such as GitHub for online recruitment. The initial version of this system is accessible on http://scsminer.wanyao.me. Figure 7 shows a screenshot of SCSMiner search system. This prototype system just gives a general overview currently, and in our future work, more functions and visualizations will be integrated.

We demonstrate SCSMiner search system mainly from two parts, the search portal part and the returned results display part. As shown in Figure 7a, in the homepage of SCSMiner search system, client users can search developers not only by entering a keyword in the input box but also by clicking the hot topic word for a quick access. After obtaining the keyword inputted by users, SCSMiner need to determine whether it is in the cache. If the keyword is



(a) Homepage of SCSMiner system **(b)** Searching results display page

Figure 7 Screenshot of SCSMiner search system

in cache, corresponding results will be returned immediately, otherwise, the model proposed in this paper will be executed, which may take about 7 seconds. In addition, we also show some statistics of GitHub dataset in the foot of this page. It's worth mentioning that 7 s is really a bit long for online service, it's necessary for us to parallelize the calculation of big matrices in the updated version.

When the ranked retrieved developers are returned, SCSMiner will jump to the result display page as shown in Figure 7b. In this page, information of developers including *name*, *email* and *gravatar* are displayed. In our future work, some other properties of developers such as their ego network will also be included. The client user can also filter the returned results by some defined conditions such as followers/following number, location, gender and so on.

7 Conclusion and future work

Social coding sites are changing software development. The developers' programming capability and expertise which conveyed by the past development experiences can benefit many applications such as software developer online recruitment for IT corporations. In this paper, we devise a generative probabilistic expertise ranking model and incorporate a consistency among projects via graph regularization. Meanwhile, a prototype of SCSMiner which provides expert search service is implemented. Comprehensive experiments conducted on the GitHub dataset show that our model performs better than the the state-of-the-art voting model and random walk model.

Besides applied in developer recruitment for IT corporation, expert finding on social coding sites can also be extended to some other areas such as question routing and developer recommendation. a) Question routing. To the best of our knowledge, most question routing approaches fall into the content-based method and graph-based method. For these two methods, one of the biggest challenge is the cold start problem. For programmers, many of them are active in both SCSs (e.g., GitHub) and Q&A sites (e.g, StackOverflow). Domain knowledge retrieved from SCS can be transferred to alleviate the cold start problem in Q&A sites. b) Developer recommendation. In collaborative development, developers can easily find a partner with specific expertise to collaborate with through our system. Moreover, if the SCSs know about each developer's expertise, it can recommend appropriate developers to other to encourage them to collaborate.

In our future work, we will first parallelize the calculation of languages models and some big matrices to accelerate the performance of SCSMiner. With the success application of deep learning in natural language processing, maybe in the future we will apply the deep learning based method such as RNN to learn the representation of users. Furthermore, we also plan to extend our model for question routing and developer recommendation based on the domain skills of developers via fusing the GitHub with StackOverflow or LinkedIn [13]. In addition, integrating data sources from multiple social coding and recruitment sites also attracts us [29, 34]. For instance, we can combine the data from GitHub and that from LinkedIn. After integrating multiple data sources, many interesting topics such as entity matching and job recommendation can be attacked.

Acknowledgments This work was down during Yao Wan's visit to University of Technology Sydney. This research was partially supported by the Natural Science Foundation of China under grant of No. 61379119 and No. 61672453, Australia Research Council Linkage Project (LP140100937). We would like to thank Lishui Zhou who helps us a lot in the implementation of the demo of SCSMiner. We would like to thank Jie Liang, Tianhan Xia and Junqing Luan for sharing their crawler source code with us and their demo system *githuber.info* also gave us some inspiration.

References

1. Balog, K., Azzopardi, L., De Rijke, M.: Formal models for expert finding in enterprise corpora. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–50. ACM (2006)
2. Balog, K., Fang, Y., de Rijke, M., Serdyukov, P., Si, L.: Expertise retrieval. Found. Trends Inf. Retr. **6**(2–3), 127–256 (2012)
3. Beeferman, D., Berger, A., Lafferty, J.: Statistical models for text segmentation. Mach. Learn. **34**(1-3), 177–210 (1999)
4. Begel, A., Bosch, J., Storey, M.-A.: Social networking meets software development: perspectives from github, msdn, stack exchange, and topcoder. IEEE Softw. **30**(1), 52–66 (2013)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
6. Buckley, C., Voorhees, E.M.: Retrieval evaluation with incomplete information. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 25–32. ACM (2004)
7. Chung, F.R.: Spectral Graph Theory, vol. 92. American Mathematical Society, Providence (1997)
8. Dabbish, L., Stuart, C., Tsay, J., Herbsleb, J.: Social coding in github: transparency and collaboration in an open software repository. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, pp. 1277–1286. ACM (2012)
9. Deng, H., Han, J., Lyu, M.R., King, I.: Modeling and exploiting heterogeneous bibliographic networks for expertise ranking. In: Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 71–80. ACM (2012)
10. Fang, H., Zhai, C.: Probabilistic Models for Expert Finding. Springer, Berlin (2007)
11. Fang, Y., Si, L., Mathur, A.P.: Discriminative models of integrating document evidence and document-candidate associations for expert search. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 683–690. ACM (2010)
12. Gyimesi, P., Gyimesi, G., Tóth, Z., Ferenc, R.: Characterization of source code defects by data mining conducted on github. In: Computational Science and Its Applications–ICCSA 2015, pp. 47–62. Springer (2015)
13. Hauff, C., Gousios, G.: Matching github developer profiles to job advertisements. In: Proceedings of the 12th Working Conference on Mining Software Repositories, pp. 362–366. IEEE Press (2015)
14. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57. ACM (1999)
15. Jiang, J., Zhang, L., Li, L.: Understanding project dissemination on a social coding site. In: 20th Working Conference on Reverse Engineering (WCRE), 2013, pp. 132–141. IEEE (2013)
16. Lima, A., Rossi, L., Musolesi, M.: Coding together at scale: github as a collaborative social network. arXiv:1407.2535
17. Macdonald, C., Ounis, I.: Voting for candidates: adapting data fusion techniques for an expert search task. In: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, pp. 387–396. ACM (2006)
18. Majumder, A., Datta, S., Naidu, K.: Capacitated team formation problem on social networks. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1005–1013. ACM (2012)
19. Marlow, J., Dabbish, L., Herbsleb, J.: Impression formation in online peer production: activity traces and personal profiles in github. In: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, pp. 117–128. ACM (2013)
20. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 275–281. ACM (1998)
21. Serdyukov, P., Rode, H., Hiemstra, D.: Modeling multi-step relevance propagation for expert finding. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 1133–1142. ACM (2008)
22. Soderland, S.: Learning information extraction rules for semi-structured and free text. Mach. Learn. **34**(1-3), 233–272 (1999)
23. Steyvers, M., Smyth, P., Rosen-Zvi, M., Griffiths, T.: Probabilistic author-topic models for information discovery. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 306–315. ACM (2004)

24. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 990–998. ACM (2008)
25. Thung, F., Bissyandé, T.F., Lo, D., Jiang, L.: Network structure of social coding in github. In: 17th European Conference on Software Maintenance and Reengineering (CSMR), 2013, pp. 323–326. IEEE (2013)
26. Vasilescu, B., Filkov, V., Serebrenik, A.: Stackoverflow and github: associations between software development and crowdsourced knowledge. In: International Conference on Social Computing (Socialcom), 2013, pp. 188–195. IEEE (2013)
27. Vendome, C., Linares-Vásquez, M., Bavota, G., Di Penta, M., German, D., Poshyvanyk, D.: License usage and changes: a large-scale study of java projects on github. In: IEEE 23rd International Conference on Program Comprehension (ICPC), 2015, pp. 218–228. IEEE (2015)
28. White, J.P.: Towards readme-eval: interpreting readme file instructions. ACL **2014**, 76 (2014)
29. Zhao, Z., Cheng, J., Wei, F., Zhou, M., Ng, W., Wu, Y.: Socialtransfer: transferring social knowledge for cold-start cowdsourcing. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 779–788. ACM (2014)
30. Zhao, Z., Yang, Q., Cai, D., He, X., Zhuang, Y.: Expert finding for community-based question answering via ranking metric network learning. In: IJCAI, pp. 3000–3006 (2016)
31. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. Advances in Neural Information Processing systems **16**(16), 321–328 (2004)
32. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 1036–1043. ACM (2005)
33. Zhou, D., Schölkopf, B.: A regularization framework for learning from graph data. In: ICML Workshop on Statistical Relational Learning and its Connections to Other Fields, vol. 15, pp. 67–68 (2004)
34. Zhu, X., Li, X., Zhang, S.: Block-row sparse multiview multilabel learning for image classification. IEEE Transactions on Cybernetics **46**(2), 450–461 (2016)