Geek Talents: Who are the Top Experts on GitHub and Stack Overflow?

Yijun Tian[1, *], Waii Ng[1], Jialiang Cao[1] and Suzanne McIntosh[1]

Abstract: In the field of Computer Science, software developers need to use a wide array of social collaborative platforms for learning and cooperating. The most popular ones are GitHub and Stack Overflow. Existing platforms only support search queries to extract relevant repository information from GitHub, or questions and answers from Stack Overflow. This ignores the valuable coder-related part-who are the top experts (geek talents) in a specific area? This information is important to companies, open source projects, and to those who want to learn from an expert role model. Thus, how to find the right developers is quite a crucial yet challenging problem. Most of the current works mainly focus on recommending experts in a particular software engineering task and ignore the relationship between developers within different projects. In this paper, we propose a novel technique that automatically identifies geek talents from GitHub, Stack Overflow, and across both communities. The results show that our work performs well at recommending proper developers in diverse areas.

Keywords: Developer recommendation, collaborative filtering, stack overflow, GitHub.

1 Introduction

Question answering (Q&A) and open source code communities have been gaining popularity in the past few years. The success of such sites depends mainly on the contribution of a small number of expert users who supply significant contributions such as helpful answers and succinct effective code. GitHub is one of the largest open source communities with more than 48 million open source projects hosted. However, according to Zhang et al. [Zhang, Wang, Yin et al. (2017)], 95.2% of them do not receive any attention from the public (i.e., no watchers or forked repositories) and 15.1% of them were not updated for more than one year. Therefore, identifying which contributors have the potential to become strong contributors is an important task which is essential for fostering enduring communities. Many expert recommendation systems [Balachandran (2013); Movshovitz-Attias, Movshovitz-Attias, Steenkiste et al. (2013); Venkataramani, Gupta, Asadullah et al. (2013); Wang, Sun, Fu et al. (2017); Yu, Wang, Yin et al. (2014); Zhang, Ackerman and Adamic (2007); Zhang, Wang, Yin et al. (2017)] have been proposed and achieve promising results since their sophisticated architectures allow them to reason about the question. To some extent, expert recommendation systems have

[1] New York University, Courant Institute of Mathematical Sciences, New York, 10012, USA.

[*] Corresponding Author: Yijun Tian. Email: yt1506@nyu.edu.

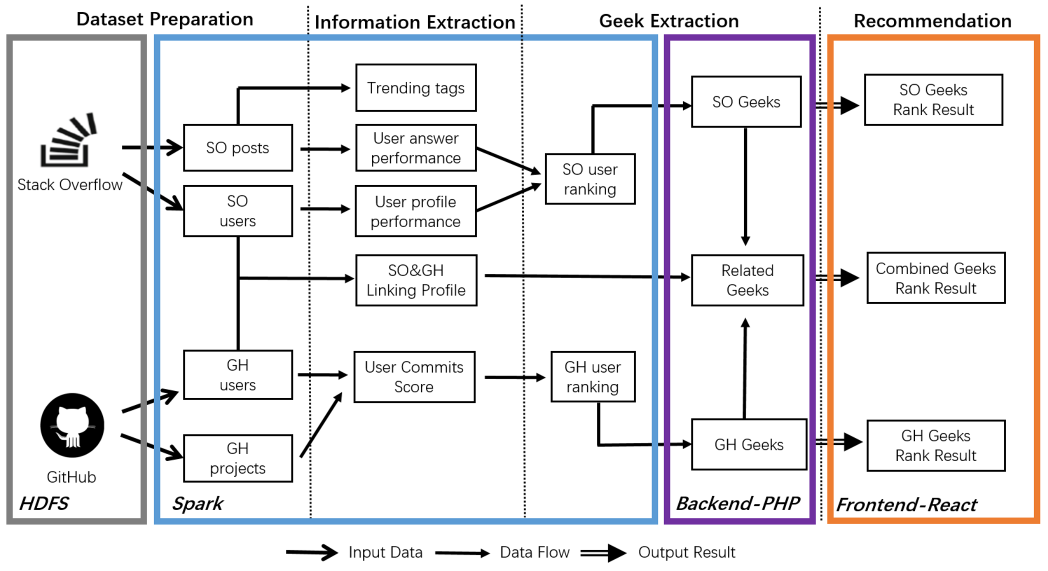


Figure 1: Pipeline of Geek Talents

shown the ability to bring great value to the open source community and to companies. Despite their success, existing expert recommendation systems mainly focus on the text data or historical information generated by users, ignoring the individual information between users.

To drive a deeper investigation into user professional activities, we are motivated to construct a cross-platform expert recommendation system matching dataset from GitHub (GH), one of the biggest code hosting sites and popular Q&A sites, to enable future studies of professional activities from multiple perspectives. Stack Overflow (SO) is the most popular Q&A community for obtaining answers to software development questions and is a rapidly growing base of information about topics ranging from algorithms to languages, with a large amount of code snippets and free-form text provided on a wide variety of fields. Vasilescu et al. [Vasilescu, Filkov and Serebrenik (2013)] shows the relationship within users between Stack Overflow and GitHub by finding GitHub users active on Stack Overflow and studying their activities on both platforms. A system as such can help us understand how different types of users (e.g., users with different expertise) are engaged in different professional activities; it can also help in understanding how different types of social interactions among users can influence the evolution of communities of different professional activities.

In this paper, we contribute a method for recommending top expert developers (geek talents) using their posted contributions to socially collaborative environments, specifically GitHub and Stack Overflow. Given any technology keyword like ‘Machine Learning’ and ‘Spark’, our recommendation system is able to extract the related top experts within the field, ranked by their liveness. Fig. 1 illustrates the pipeline of our recommendation approach. By exploiting different attributes of user profiles, platform-specific APIs, and a variety of account matching strategies, there are four key parts in our proposed method, including data preparation, information extraction, geek extraction, and recommendation. Data Preparation reconstructs and cleans the coarse data, to generate elaborated data with the required information. Then, Information Extraction is used to filter the valuable information including the relationship graph between users, posts, and repositories.

Information streams are transferred within the same data source. After that, Geek Extraction is used to extract SO (Stack Overflow) geeks as well as GH (GitHub) geeks using the SO-based and GH-based approaches. Related geeks are generated by joining them together with an effective selection method. Finally, from the geeks we extracted above, a visualization provides our users with an intuitive view of geek talents in a given field of interest.

By characterizing the network features, we present our recommendation ranking result based on how users interact with others in the same field, and how different activities of the same user correlate with each other. Since GitHub only fetches hot projects given one query, our work shows great importance for its novelty and convenience.

The main contributions of this paper are summarized as follows:

- We propose a novel schema that automatically finds geek talents in a specific field from GitHub, Stack Overflow and across the two platforms.
- We derive a new method to deal with the user extraction problem, consisting of a SO-based approach, a GH-based approach, as well as an approach to combine them with a particular weighting factor.
- We build a carefully designed user interface that visualize the result, which makes the exploration of large, complex user data an easier job.

2 Motivation

Modern software development depends heavily upon cooperation between developers to increase productivity and reduce time-to-market. Many popular libraries and frameworks have presented strategies to increase the on-boarding as well as engagement of new contributors, and developers tend to accomplish the work jointly. In this situation, each person is only responsible for part of it, no need to have a full understanding of the whole software system. Thus, a platform that provides source code management and distributed version control collaboration is required. The most famous one is GitHub, which supports bug tracking, feature requests, task management, and wikis for every project.

However, most of the platforms only support searching for query related code repositories; they lack the capability to extract or recommend influential users in a specific field. Nevertheless, knowing top experts has a practical value. For example, an open source project manager can use this information to find potential contributors. Private companies can employ it to hire suitable employees. In addition, by following those experts in social collaborative platforms like GitHub and Stack Overflow, beginners can get a quick and thorough comprehension of the cutting-edge knowledge in this field. The deep insight and successful learning path exposed by following experts makes the learning process much easier and saves time. In this context, finding experts among the members of global open-source software development platforms is critical.

3 Related work

3.1 General expert recommendation

Several studies of technical support communities have measured and modeled the expertise of community members.

Zhang et al. [Zhang, Ackerman and Adamic (2007)] incorporate the underlying interaction dynamics to investigate the Java Forum, a help-seeking community for Java programmers and to extract users with high expertise, by using social network analysis methods like PageRank [Page, Brin, Motwani et al. (1998)] and HITS [Kleinberg (1999)]. The comparison between alternative algorithms for ranking expertise shows that structural information and social network-based algorithms can be used for evaluating an expertise network.

Xuan et al. [Xuan, Jiang, Ren et al. (2012)] addressed the problem of developer prioritization, which aims to rank the contributions of developers by exploring two aspects, modeling the developer prioritization in a bug repository and incorporating it to improve predictive tasks.

Balachandran [Balachandran (2013)] proposed a reviewer recommendation algorithm to ease the task of finding appropriate reviewers in a large project.

Wang et al. [Wang, Sun, Fu et al. (2017)] and Mao et al. [Mao, Yang, Wang et al. (2015)] investigated the problem of recommending skilled developers to work on programming tasks posted on the TopCoder crowdsourcing platform.

Fazel-Zarandi et al. [Fazel-Zarandi, Fevlin, Huang et al. (2011)] presented an expert recommender system capable of applying multiple theoretical mechanisms to the problem of personalized recommendations through profiling users' motivations and their relations.

3.2 GitHub expert recommendation

Social work environments such as GitHub make the relationships between users transparent. Thus, many studies have been presented to understand the influence of users and to find the potential influential developers.

Tsay et al. [Tsay, Dabbish and Herbsleb (2014)] evaluated potential contributors to open source software projects by pull requests (PRs), which are one of the primary methods for contributing code in GitHub.

Yu et al. [Yu, Wang, Yin et al. (2014)] analyzed the PR mechanism, and proposed a reviewer recommender to predict highly relevant reviewers of incoming PRs.

Montandon et al. [Montandon, Silva and Valente (2019)] proposed a method to identify library experts based on GitHub by clustering feature data.

Liao et al. [Liao, Jin, Li et al. (2017)] proposed DevRank, which ranks developers by influence propagation through a heterogeneous network according to user behavior, including “commit” and “follow”.

Hu et al. [Hu, Wang, Ren et al. (2018)] proposed a Following-Star-Fork-Activity based approach to measure user influence in the GitHub developer social network.

Thongtanunam et al. [Thongtanunam, Tantithamthavorn, Kula et al. (2015)] proposed RevFinder, a file location-based code-reviewer recommendation approach to recommend appropriate code-reviewers for developers.

3.3 Stack Overflow expert recommendation

Since Stack Overflow is one of the biggest online Question Answer communities, which generates huge user content continuously, it has been used for research analysis to find top experts in a specific domain [Wang, Huang, Yao et al. (2018)]. Among which, reputation score is a valuable impact factor that many methods have taken advantage of.

Bosu et al. [Bosu, Corley, Heaton et al. (2013)] investigated the dynamics of reputation scores and various attributes involved in assignment from four perspectives. Sumanth et al. [Sumanth and Rajeshwari (2018)] tried to identify top experts based on the reputation score, HITS score, and PageRank algorithm.

Movshovitz-Attias et al. [Movshovitz-Attias, Movshovitz-Attias, Steenkiste et al. (2013)] considered a number of graph analysis methods for detecting influential and anomalous experts in the underlying user interaction network. However, reputation scores calculated by Stack Overflow are irrespective of specific domain and are restricted to the internal platform and not available to the public.

Yang et al. [Yang, Tao, Bozzon et al. (2014)] proposed a novel metric for expert identification in Stack Overflow by describing several behavioral properties and focusing on the quality of their contributions.

Riahi et al. [Riahi, Zolaktaf, Shafiei et al. (2012)] investigated the suitability of two statistical topic models and compare these methods against more traditional Information Retrieval approaches to recommend experts for a newly posted question.

3.4 Cross platform expert recommendation

Diverse social platforms enable the distinctive opportunity for understanding the user from multiple aspects. By combining cross platform user profiles and incorporating various user activity information, much research has been performed to find correlation, and based on this, to recommend experts.

Chen et al. [Chen and Xing (2016)] analyzed the searches in Google and Stack Overflow communities. They tried to find out correlations between the search keyword in Google, in fetching a Stack Overflow result.

Silvestri et al. [Silvestri, Yang, Bozzon et al. (2015)] conducted a comparative study to investigate the correlations between user interactions across Stack Overflow, GitHub and Twitter.

Venkataramani et al. [Venkataramani, Gupta, Asadullah et al. (2013)] built a recommendation system for Stack Overflow that use the data mined from GitHub.

Constantinou et al. [Constantinou and Kapitsaki (2016)] measured developers' commit activity on GitHub by considering both the quantity and the continuity of their contributions in isolated projects through time, and evaluated it with the answering activity in Stack Overflow.

Huang et al. [Huang, Mo, Shen et al. (2016)] proposed CPDScorer that scores developers' skills by correlating developer activity on Stack Overflow and GitHub.

Zhang et al. [Zhang, Wang, Yin et al. (2017)] proposed a recommendation system DevRec on GH and SO based on an association matrix. However, much information within the user attributes are not considered and the weighting influence presented in the information stream is ignored.

4 Expert recommendation algorithms

4.1 Stack Overflow expert recommendation algorithm

In Stack Overflow, we first extract trending tags information and user answer performance information from the SO *posts* dataset. Then, given a specific tag key word, we combine the related user answer performance and user profile performance extracted from SO *users* dataset, to produce SO user influential factor. Then, by sorting the SO user influential factor, the SO user ranking result is generated.

In the extraction of trending tags information, we use the questions within the SO posts dataset to evaluate current trending topics. We record the tag of each question, indicate the viewed number $viewCount_{post}$ provided by SO as the frequency of a specific post and calculate the total frequency of each tag. Finally, we filter the trending tag set Set_{Tag} by extract those that appeared more than 1000 times, which is defined as

$$viewCount_{tag}>1000,\ \forall tag\in Set_{Tag} \tag{1}$$

$$viewCount_{tag}=\sum_{i=1}^{postNum_{tag}} viewCount_{posti} \tag{2}$$

where $viewCount_{tag}$ is the total frequency of each tag, and where $postNum_{tag}$ indicates the number of existing posts in the related tag.

In the calculation of SO user influential factor, we use S_{userSO} to indicate the SO user influential score, which is defined as

$$S_{userSO}=0.6\times S_{UAP}+0.4\times S_{UPP} \tag{3}$$

where S_{UAP} denotes the user answer performance score and S_{UPP} is the user profile performance score.

In the extraction of user answer performance information, we indicate the answer post with user ID $User_{id}$ and related answer score S_{ans}. We indicate the question post as tetrad (*tag, viewCount*$_{post}$*, favoriteCount, ansCount*), where *tag* denotes the tag name, *favoriteCount* is the number of times this post was favorited by people and *ansCount* is the related number of answers under this post. Then we concatenated the question post with the corresponding answer post to generate one to one correspondence. Then we calculate the user answer performance score S_{UAP} as

$$S_{UAP}=0.5\times S_{ans}+0.5\times S_{question} \tag{4}$$

$$S_{question}=0.3\times avgViewCount$$

$$+0.3 \times avgFavoriteCount \quad (5)$$
$$+0.3 \times avgAnsCount$$

Here, S_{ans} is the average score calculated from the answer posts and $S_{question}$ indicates the average score calculated from question posts. In Eq. (5), *avgViewCount*, *avgFavoriteCount*, *avgAnsCount* denotes the average number of *viewCount*, *favoriteCount*, *ansCount* per question.

In the extraction of user profile performance information, we use the weighted sum of user reputation *reputation* and user profile view count $viewCount_{user}$ to generate the user profile performance score S_{UPP}, which is defined as

$$S_{UPP}=0.7\times reputation+0.3\times viewCount_{user} \quad (6)$$

4.2 GitHub expert recommendation algorithm

Since we are trying to recommend talented users under specific tags, which is not supported by GitHub currently, we propose a novel method to bypass this limitation. By incorporating the repository and the related contributor information, we are able to find influential users under a specific tag key word.

First, we use the GitHub search API to extract the related repositories under a particular tag word. Then, for each user who committed to one repository, we calculate his or her score towards this repository according to the number of commits he or she made, and the repositories per commit score. After that, we sum up and get each user's total score S_{userGH}, which is defined in formula 7. Finally, we rank the users by this score to get GitHub users ranking result for a specific searching tag key word.

$$S_{userGH} = \sum_{repo=0}^{repoNum} S_{repo,user} \quad (7)$$

where *repo* is the index of repository, *repoNum* indicates the number of existing repositories in a related tag, *user* represents the same user as *userGH*. $S_{repo,user}$ is the contributing score the *user* possesses relative to a specific repository *repo*, which is defined as

$$S_{repo,user} = commits_{repo,user} \times S_{repo,PC} \quad (8)$$

where $commit_{repo,user}$ represents the commit frequency for the *user* contributing to the repository *repo*, and $S_{repo,PC}$ indicates the score of per commit (PC) for the repository *repo*, which is defined as

$$S_{repo,PC} = Weight_{tag} \times (watchers_{repo} / commits_{repo}) \quad (9)$$

where $Weight_{tag}$ represents the weight factor of this repository under the specific *tag*, which is defined in formula 10. $watchers_{repo}$ indicates the number of watches received in the repository *repo*. $commits_{repo}$ demonstrates the number of commits received in the repository *repo*.

$$Weight_{tag} = BOC_{tag} / BOC_{total} \quad (10)$$

Here, the direct explanation of $Weight_{tag}$ is the tag related language percentage in the repository, where BOC_{tag} is the byte of code (BOC) corresponding to tag related type and BOC_{total} is the total byte of code in the whole repository.

4.3 Combined expert recommendation algorithm

As described in Fig. 1, we combined SO users and GH users to generate SO and GH linking profile in the information extraction section. There are several ways to combine the users. The most common way is to link them with email addresses. Here, we consider two ways, using email addresses and using GH username.

For the former one, since the SO *users* dataset already provides the processed email address hashed by MD5 message-digest algorithm, we only need to hash GH user email addresses with the same hash function. Then, we take the hashed email as key and link the accounts with the same hash value of GH email address and SO email address. In linking with GH username, the most difficult issue to resolve is how to find the corresponding GH username for a given SO user. To deal with this challenge, we extract GH username for each user in SO from the SO user description, which only appears in the section named *About Me* in the SO *users* dataset.

5 Datasets

5.1 GitHub

We use three different kinds of APIs to access the GitHub dataset: The *Search API*, the *Users API*, and the *Repository API*. The *Search API* enables us to search for the specific projects we want to find. Given one interesting search query, it can provide up to 1,000 results at a time. Here results are sorted by best match, as indicated by the score field within each item returned. This is a computed value representing the relevance of an item relative to the other items in the result set. Multiple factors are combined to boost the most relevant item to the top of the result list.

We use the *Users API* to get information about the currently authenticated user. It includes followers and following information. Also it can help us with extracting useful contents like events and repositories. By using this information, we can build a graph between users. For the primary key setting, we are trying to use username and related email address. The *Repository API* facilitates access to repositories a user owns, repositories they contribute to, and repositories that they can access through an organization membership. Therefore, we can find user information through given related hot repositories. We can use it to list the topics in a repository, list contributors, and list user repositories.

5.2 Stack Overflow

The Stack Overflow dataset used in the project contains posts of questions and answers related to various domains that were asked on the Stack Overflow community. With the aim of making our work more convincing, we take hashed emails as one of our matching accounts strategy. Due to privacy concern, the end of 2014 Stack Overflow data dump no longer contains hashed user emails. To overcome this limitation, we adopt the data dump which offers hashed email address (released on August 2012) and extended it with the latest *users* data contained in March 2019 data dump (which is the last released dump so far). For the *posts* dataset, we just use the latest one, which is also published in March 2019. The dataset is made available online by Stack Overflow on their official website in the xml format for usage.

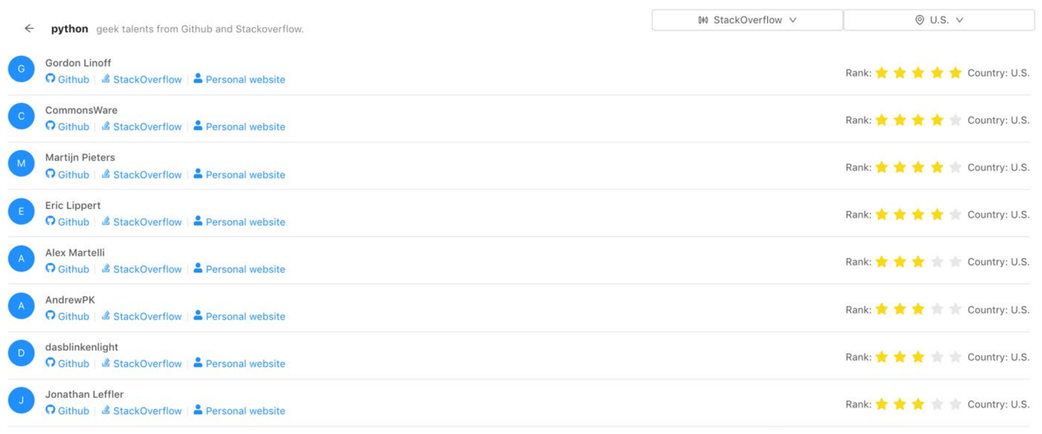
5.3 Attribute selection on each dataset

With the aim of identifying geek talents, we take several datasets into account. Each dataset has different but meaningful attributes.

Tab. 1 shows the attributes we selected and the related content types within each dataset. In GitHub data, we mainly use the *users* table which includes personal information registered on GitHub. Since the projects that users have worked on can indicate their interests in terms of software projects, we also build a projects table to include repository information. Similarly, in Stack Overflow data, we use the *users* table as well as *posts* to extract top experts.

Table 1: Selected attributes and content types for each dataset

Dataset	Table	Attribute	Type
Stack Overflow	***Users***	userID	Int
		$\text{viewCount}_{\text{user}}$	Int
		reputation	Int
		displayName	String
		location	String
		websiteUrl	String
		aboutMe	String
		hashedEmail	String
	posts	postID	Int
		acAnswerID	Int
		parentID	Int
		postType	Int
		answerScore	Int
		favoriteCount	Int
		$\text{viewCount}_{\text{post}}$	Int
		creationDate	date
		tag	String
GitHub	***Users***	userID	Int
		commits	Int
		userName	String
		Email	String
		countryCode	String
	projects	watchers	Int
		commits	Int
		bytes	Int
		language	String
		topics	String
		labels	String



python geek talents from Github and Stackoverflow.

StackOverflow | U.S.

Gordon Linoff
Github | StackOverflow | Personal website
Rank: ★★★★★ Country: U.S.

CommonsWare
Github | StackOverflow | Personal website
Rank: ★★★★☆ Country: U.S.

Martijn Pieters
Github | StackOverflow | Personal website
Rank: ★★★★☆ Country: U.S.

Eric Lippert
Github | StackOverflow | Personal website
Rank: ★★★★☆ Country: U.S.

Alex Martelli
Github | StackOverflow | Personal website
Rank: ★★★☆☆ Country: U.S.

AndrewPK
Github | StackOverflow | Personal website
Rank: ★★★☆☆ Country: U.S.

dasblinkenlight
Github | StackOverflow | Personal website
Rank: ★★★☆☆ Country: U.S.

Jonathan Leffler
Github | StackOverflow | Personal website
Rank: ★★★☆☆ Country: U.S.

Figure 1: UI for Geek Talents

In order to find geek talents, combinations of attributes are determined in a heuristic manner, and weights for the combinations are calculated via the particular scheme shown in Section 4. Also, we combine users from both GitHub and Stack Overflow to provide a cross platform recommendation. However, different datasets may have their own description for the user-related attributes. Thus, we correlate different attributes by a specific matching method across Stack Overflow and GitHub. The corresponding relationship can be found in Tab. 2.

Table 2: Mapping between Stack Overflow and GitHub

Attributes on Stack Overflow	Attributes on GitHub
users.userID	users.userID
users.displayName	users.userName
users.location	users.countryCode
users.hashedEmail	users.Email
post.tag	projects.language

6 Application design

6.1 Front-end design

The Geek-Talents web application is implemented with React. There are two main React components, Tags Table and User List, which fetch data through APIs and show visualizations to users. By fetching trending tags through the Tags API provided by the back-end, the Tags Table can visualize those topics with trending rate, while for the user list, once the user selects a topic from topic table, this component fetches top rated users for the given tag and visualizes the ranking results with user profile. Fig. 2 shows an example of the visualization interface. A list format of page with personal website, GitHub user profile and Stack Overflow user profile has been generated and contained. Filtering options such as country and ranking source (e.g., GitHub ranking, Stack Overflow ranking, combined ranking of GitHub and Stack Overflow) are also provided.

6.2 Back-end design

We employed MySQL as our Back-end data platform and organized the information into four tables: *SO user table*, *GH user table*, *MIX user table*, and *tags table*. *SO user table* contains the SO users score S_{userSO} for each tag and account information, *GH user table* contains the GH users score S_{userGH} for each tag and account information, *MIX user table* contains linked SO & GH users combined score for each tag and information and *tags table* contains the recent trending tags. We used a PHP server to process queries coming from the front-end, to analyze the query URL, and to extract keywords from the query. Since the PHP server is connected to the MySQL database, it can retrieve the sorted data from the database and return it to the front-end according to the keywords within the query.

To better communicate with the front-end, we created convenient APIs in the back-end, including Users API and Tags API. For the former one, the user information can be simply extracted by using "/users?Tag=[*tag*] & CountryCode=[*countryCode*] & Source=[*so/gh/mix*]", where *tag* stands for the technical tag in search. This value will be HTML encoded in case special characters appear (e.g., "c++" will be encoded as "c%2B%2B"). *countryCode* stands for ISO 3166-1 alpha-2 which are two-letter country codes ("United States" will be written as "us"). The server will return all eligible results regardless of country information if this value is empty in the query. The choices for source, *so/gh/mix*, represent retrieving information from Stack Overflow, GitHub, or both. The extracted result of the API query is formed in JSON format and sorted in descending order by "Rank", which is shown in Fig. 3.

```
{
  "SO_Id":"123",
  "GH_Id":"234",
  "SO_UserName":"Bob",
  "GH_UserName":"Bob",
  "Tag":"java",
  "Rank":"3.8",
  "WebsiteUrl":"https://github.com",
  "HashedEmail":"11efb70ba7f5a52d323fc540468435b1",
  "CountryCode":"us"
}
```

Figure 2: Example return result from back-end Users API under query "/users? Tag=[java] & CountryCode=[us] & Source=[mix]"

```
{
  "_Tag":"Python",
  "_ViewCount":"129254"
}
```

Figure 3: Example return result from back-end tags API under query "/tags? Tag=[Python]"

With the Tags API, we can extract related information with the query "/tags?Tag=[*tag*]". An example return result has been shown in Fig. 4, which is formed in Json format and sorted in descending order by "viewCount".

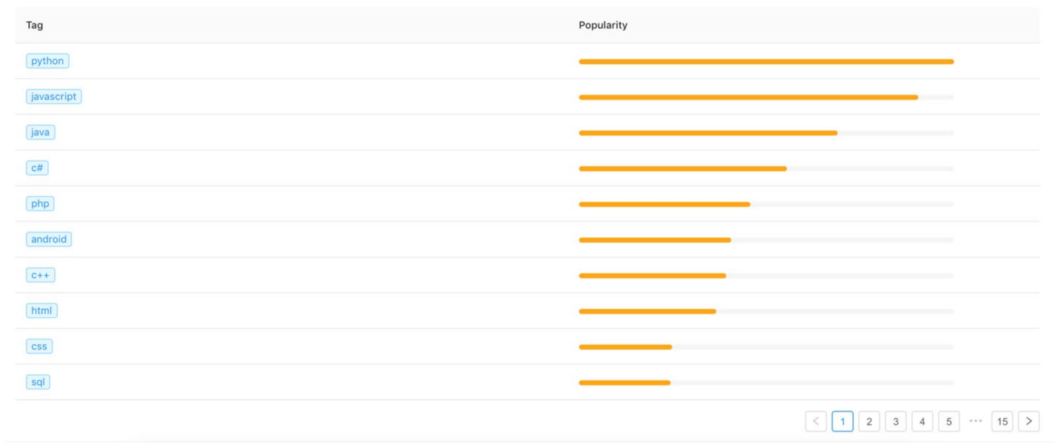


Figure 4: Visualization of popular tags

7 Experiments

In order to generate the SO and GH linking profile information, we need to first extract user information from SO and GH. We extracted 1,295,622 users from the SO *users* dataset and 7,953,512 users from the GH users dataset. With the method we mentioned in Section 4.3, we linked 332,362 users for our combined expert recommendation, including 309,735 using the hashed email address method and 28,294 using the GH username method. Both of them contain SO 'question and answer' information as well as GH 'activity and coding' information.

For the choice of tag in GitHub, there are several options. The first one is filtering by topics in GitHub which were launched in recent years. We followed the process described in section 4. However, finding a repository with labeled topics is rare. Thus, we only found 56,889 pairs of (*user, tag*) tuples. Next, we tried to use the label from the repository as the searching tag. Still, the results are sparse and inaccurate since 1) labels are defined by the user arbitrarily without consistent standards and 2) the correlation between labels and repository content is unconsolidated. Finally, we incorporated the language tag generated by GitHub for each repository, which overcomes all of the limitations described before with abundant, accurate, and consistent descriptions. As a result, 2,548,505 pairs of (*user, tag*) tuples have been generated.

After we selected the language tags (for programming language) provided by GitHub as our searching key word, we found that, while the talented user recommended by our system did have the language tag in their information, sometimes the ranking may not be accurate. Some influential experts in the tag language field are beaten by those who only use this language a few times in the repository. It demonstrates that our system calculates repository scores equally regardless of the proportion of tagged languages, and the recommended users often use the language as a supplement rather than as the main language, resulting in very high scores for other languages with very small representation (e.g., developers always use CPP language in the underlying algorithms to reduce complexity, while the other algorithms use Python for the conciseness and convenience).

As a result, we take $Weight_{tag}$ described in formula 10 into account, which successfully allows the system to pay more attention to the main language of a given repository.

As described in Section 4.3, only combined users have the information from both the SO and GH sides. However, when it comes to GH users and SO users, they can only access their own information, which limits the integrity of the information for recommended user when searching under a single platform. Therefore, we use left outer join method to match the combined user again, so that the complete information of both websites can be viewed from both SO users and GH users.

In order to support the feature of regional selection shown in Fig. 4, we first try to classify users by different cities. There are 106,263 different cities in the SO set, 31,205,585 different cities in the GH set and 18,858 same cities between SO and GH. Even after normalization, the city names are still at a massive scale. Therefore, classifying users by country has been taken into consideration. Since we already have 249 different clean country codes from the GH users set, we only need to normalize the location in the SO users set. In addition, we use ISO-3166-Countries-with-Regional-Codes table to map the relationship between SO location and GH country code.

For the choice of trending topic tags, we extracted the latest month post tags from SO post dataset and found 12,370 different tags. In order to filter out the post tags that are rarely used, we only take those tags that appeared more than 1,000 times within a month into account. Then, we connect those tags with GH tags after data cleaning. The reason we select SO post tags rather than GH project tags to begin with mapping is 1) the division between different topics are more specific and 2) we believe the questions that appear in new technologies are always faster and easier to appear than projects in new technologies.

8 Conclusion

In summary, in this paper we addressed the problem of expert recommendation in GitHub, Stack Overflow, and across both platforms. We proposed a novel methodology to deal with the user extraction problem, which makes full use of different user attributes and related platform specific information. Furthermore, a platform with a carefully designed user interface has been built with the aim of visualization, which makes the exploration of large, complex user data an easier job.

We hope our work can inspire future work across other social collaboration platforms. Also, since our system assumes that developers with high quality Stack Overflow answers (measured by number of upvotes) are more likely to be experts in specific programming technologies; the same is assumed for developers who contributed to high quality projects, as measured using source code metrics, more specific prerequisites can be set to produce better performance.

Acknowledgment: We thank NYU High Performance Computing (HPC) for their support and Cloudera for providing the CDH Hadoop distribution through the Cloudera Academic Partnership.

References

Balachandran, V. (2013): Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. *35th International Conference on Software Engineering*.

Bosu, A.; Corley, C. S.; Heaton, D.; Chatterji, D.; Carver, J. C. et al. (2013): Building reputation in Stackoverflow: an empirical investigation. *10th Working Conference on Mining Software Repositories*.

Chen, C.; Xing, Z. (2016): Towards correlating search on google and asking on stack overflow. *IEEE 40th Annual Computer Software and Applications Conference*.

Constantinou, E.; Kapitsaki, G. M. (2016): Identifying developers' expertise in social coding platforms. *42th Euromicro Conference on Software Engineering and Advanced Applications*.

Fazel-Zarandi, M.; Devlin, H. J.; Huang, Y.; Contractor, N. (2011): Expert recommendation based on social drivers, social network analysis, and semantic data representation. *2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems.*

Hu, Y.; Wang, S.; Ren, Y.; Choo, K. R. (2018): User influence analysis for GitHub developer social networks. *Expert Systems with Applications*, vol. 108, pp. 108-118.

Huang, W.; Mo, W.; Shen, B.; Yang, Y. K.; Li, N. (2016): CPDScorer: modeling and evaluating developer programming ability across software communities. *28th International Conference on Software Engineering and Knowledge Engineering.*

Kleinberg, J. M. (1999): Authoritative sources in a hyperlinked environment. *Journal of the ACM*, vol. 46, no. 5, pp. 604-632.

Liao, Z.; Jin, H.; Li, Y.; Zhao, B.; Wu, J. et al. (2017): Devrank: mining influential developers in GitHub. *IEEE Global Communications Conference.*

Mao, K.; Yang, Y.; Wang, Q.; Jia, Y.; Harman, M. (2015): Developer recommendation for crowdsourced software development tasks. *IEEE Symposium on Service-Oriented System Engineering*.

Montandon, J. E.; Silva, L. L.; Valente, M. T. (2019): Identifying experts in software libraries and frameworks among GitHub users. arXiv.org/abs/1903.08113.

Movshovitz-Attias, D.; Movshovitz-Attias, Y.; Steenkiste, P.; Faloutsos, C. (2013): Analysis of the reputation system and user contributions on a question answering website: stack overflow. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*.

Page, L.; Brin, S.; Motwani, R.; Winograd, T. (1998): The PageRank citation ranking: bringing order to the web. *Stanford Digital Libraries Working Paper*, vol. 9, no. 1, pp. 1-14.

Riahi, F.; Zolaktaf, Z.; Shafiei, M.; Milios, E. (2012): Finding expert users in community question answering. *Proceedings of the 21st International Conference on World Wide Web*, vol. 791.

Silvestri, G.; Yang, J.; Bozzon, A.; Tagarelli, A. (2015): Linking accounts across social networks: the case of Stack Overflow, GitHub and twitter. *1st International Workshop on*

Knowledge Discovery on the WEB.

Sumanth, P.; Rajeshwari, K. (2018): Discovering top experts for trending domains on Stack Overflow. *8th International Conference on Advances in Computing Communications*, pp. 333-340.

Thongtanunam, P.; Tantithamthavorn, C.; Kula, R. G.; Yoshida, N.; Iida, H. et al. (2015): Who should review my code? A file location-based code-reviewer recommendation approach for modern code review. *IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering*.

Tsay, J.; Dabbish, L.; Herbsleb, J. (2014): Influence of social and technical factors for evaluating contribution in GitHub. *Proceedings of the 36th International Conference on Software Engineering*, pp. 356-366.

Vasilescu, B.; Filkov, V.; Serebrenik, A. (2013): Stackoverflow and GitHub: associations between software development and crowdsourced knowledge. *International Conference on Social Computing*.

Venkataramani, R.; Gupta, A.; Asadullah, A.; Muddu, B.; Bhat, V. (2013): Discovery of technical expertise from open source code repositories. *Proceedings of the 22nd International Conference on World Wide Web*, pp. 97-98.

Wang, X.; Huang, C.; Yao, L.; Benatallah, B.; Dong, M. (2018): A survey on expert recommendation in community question answering. *Journal of Computer Science and Technology*, vol. 33, no. 4, pp. 625-653.

Wang, Z.; Sun, H.; Fu, Y.; Ye, L. (2017): Recommending crowdsourced software developers in consideration of skill improvement. *32nd IEEE/ACM International Conference on Automated Software Engineering*.

Xuan, J.; Jiang, H.; Ren, Z.; Zou, W. (2012): Developer prioritization in bug repositories. *34th International Conference on Software Engineering*, pp. 25-35.

Yang J.; Tao, K.; Bozzon, A.; Houben, G. J. (2014): Sparrows and owls: characterisation of expert behaviour in StackOverflow. *22nd Conference on User Modeling, Adaptation and Personalization.*

Yu, Y.; Wang, H.; Yin, G.; Ling, C. X. (2014): Reviewer recommender of pull-requests in GitHub. *IEEE International Conference on Software Maintenance and Evolution*.

Zhang, J.; Ackerman, M. S.; Adamic, L. (2007): Expertise networks in online communities: structure and algorithms. *Proceedings of the 16th International Conference on World Wide Web*.

Zhang, X.; Wang, T.; Yin, G.; Yang, C.; Yu, Y. et al. (2017): Devrec: a developer recommendation system for open source repositories. *Mastering Scale and Complexity in Software Reuse*.