International Journal of Software Engineering
and Knowledge Engineering
Vol. 27, Nos. 9&10 (2017) 1409–1425

DOI: 10.1142/S0218194017400034

Developer Identity Linkage and Behavior Mining Across GitHub and StackOverflow

Yunxiang Xiong[*,‡], Zhangyuan Meng[*,§], Beijun Shen[*,¶] and Wei Yin[†,‖]

[*]*School of Software*
Shanghai Jiao Tong University
Shanghai 200240, P. R. China

[†]*China Aeronautical Radio Electronics Research Institute*
Shanghai 200233, P. R. China
[‡]*bruinx@sjtu.edu.cn*
[§]*602389789@sjtu.edu.cn*
[¶]*bjshen@sjtu.edu.cn*
[‖]*yin_wei@careri.com*

Nowadays, software developers are increasingly involved in GitHub and StackOverflow, creating a lot of valuable data in the two communities. Researchers mine the information in these software communities to understand developer behaviors, while previous works mainly focus on mining data within a single community. In this paper, we propose a novel approach to developer identity linkage and behavior mining across GitHub and StackOverflow. This approach links the accounts from two communities using a CART decision tree, leveraging the features from usernames, user behaviors and writing styles. Then, it explores cross-site developer behaviors through T-graph analysis, LDA-based topics clustering and cross-site tagging. We conducted several experiments to evaluate this approach. The results show that the precision and F-score of our identity linkage method are higher than previous methods in software communities. Especially, we discovered that (1) active issue committers are also active question askers; (2) for most developers, the topics of their contents in GitHub are similar to those of those questions and answers in StackOverflow; (3) developers' concerns in StackOverflow shift over the time of their current participating projects in GitHub; (4) developers' concerns in GitHub are more relevant to their answers than questions and comments in StackOverflow.

Keywords: Identity linkage; developer behavior mining; GitHub; StackOverflow.

1. Introduction

In recent years, software developers are intensively involved in open-source software development communities (e.g. GitHub) and knowledge-sharing communities (e.g. StackOverflow). As developers continuously contribute to or exchange ideas through these communities, a lot of development data and knowledge are accumulated there.

¶Corresponding author.

According to the data from ghtorrent[a] and archive.org,[b] there are more than 30 million repositories and 6 million developers in GitHub; and 40 million posts and 8 million users in StackOverflow as of August 2017. It is a great opportunity to understand working habits and patterns of software developers through analyzing and mining these data.

In previous studies on data mining of software communities, researchers focused more on a single community. For example, Robinson *et al.* [1] studied on the developer behavior and sentiment from data mining open-source repositories. Kouters *et al.* [2] tried to understand the dynamic issue in GitHub project. Treude *et al.* [3] researched about how do programmers ask and answer questions on StackOverflow. Bird *et al.* [4] explored a question: whether the programming skills and knowledge are related to age. It will benefit more to mine developer behavior across software communities, which can give us a deeper understanding of the various behaviors involved in the software development process from different perspectives. However, very few works link the different communities and do analysis and mining across multi-communities: Geominne and Mens [5] linked StackOverflow to Issue Tracker for issue resolution but they did not find the association between the two platforms. Ahmed and Srivastava [6] tried to find some associations between software development and crowdsourced knowledge, but the result they gave is too simple and incomplete.

Generally, we are facing three challenges when mining developer behaviors across software communities:

(1) Identity linkage problem: The traditional social network relies on usernames and email addresses for linking identities between communities. However, StackOverflow has no longer provided users' email addresses. Thus it lacks strong evidence for linking users with the same identities between GitHub and StackOverflow.
(2) Data heterogeneous problem: Data across different software communities is heterogeneous. For example, labels of repositories in GitHub are programming languages, but those of questions and answers (Q&As) in StackOverflow are technical terms. We can get the developer age data on GitHub but cannot get the data on StackOverflow.
(3) Association mining: After identity linkage, it is also challenging to find the associations of developer behaviors across GitHub and StackOverflow, and recover the latent, valuable information of these data.

To address these challenges, this paper proposes a novel approach for mining developer behaviors across GitHub and StackOverflow, as shown in Fig. 1. It consists of two phases: identity linkage and behavior mining.

At the identity linkage phase, we extract the features from the developer profile and behavior data, including the similarity between usernames, user behaviors and

[a] http://www.ghtorrent.org.
[b] https://archive.org/download/stackexchange.

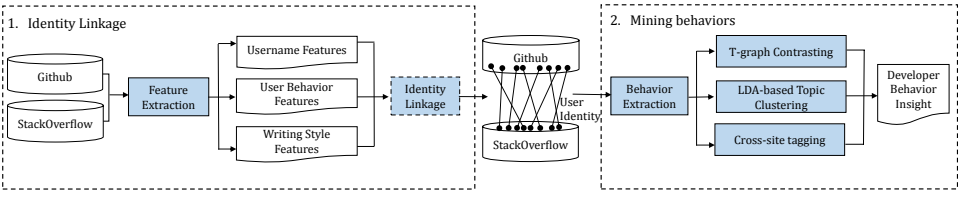

Fig. 1. Approach overview.

user writing styles. And then classification and regression tree (CART) algorithms are applied to link the accounts of developers between GitHub and StackOverflow.

At the behavior mining phase, we raise three research questions for exploring the patterns on developer behaviors across these two software communities. Statistics, Natural Language Processing (NLP) and machine learning technologies are adapted to analyze and mine the merged developer behavior data.

Our main contributions are summarized as follows: We propose an approach for mining cross-site developer behaviors. It links identities between GitHub and StackOverflow by leveraging features from usernames, user behaviors and writing styles, using CART decision tree. And then it mines the merged developer behavior data to find some valuable observations. We conducted several experiments to evaluate the mining approach. The results show that the precision and F-score of our identity linkage method are higher than previous methods in software communities. Especially, we discovered that (1) active issue committers are also active question askers; (2) for most developers, the topics of their contents in GitHub are similar to those of their questions and answers in StackOverflow; (3) developers' concerns in StackOverflow shift over the time of their current participating projects in GitHub; (4) developers' concerns in GitHub are more relevant to their answers than questions and comments in StackOverflow.

The remainder of this paper is organized as follows. After reviewing the related works in Sec. 2, Section 3 focuses on the identification of the same user in both communities. It describes the extraction and preprocessing of data, the identification of the same identity and the process and results of the experiment. In Sec. 4, we describe the preparation of data for mining, and present four problems and the reasons for them. The methods, processes and findings are also given in this section. Finally, conclusions and future work are offered in Sec. 5.

2. Related Works

2.1. *Identity linkage across software communities*

Several methods have been proposed to solve the identity linkage problem in software communities. A simple algorithm [5] was proposed by Goeminne and Mens using several simple rules to judge if two user pairs are the same person. Bird *et al.* [4] proposed a more advanced algorithm in which some text similarity metrics are used,

such as Levenshtein distance. Furthermore, a semantic-based method LSA was proposed to link users by Kouters *et al.* [2]. However, these methods did not use username as the most important one of the features to improve prediction accuracy, nor did they take full advantage of the textual information left by developers in software development.

2.2. *Developer behavior mining in software communities*

There are many researches focusing on mining the data from a single community. For example, Robinson *et al.* [1] showed some developer behaviors and sentiments from open-source repository mining; Treude *et al.* [3] argued about how do programmers ask and answer questions on StackOverflow. On the other hand, few people studied the association between these two communities. Vasilescu *et al.* [7] investigated the interplay between StackOverflow activities and the development process in GitHub. They found that the QA activity rate correlates with the code changing activity. So, across software communities, there are still a lot of patterns and insights to explore.

3. Identity Linkage Across GitHub and StackOverflow

Accounts from GitHub and StackOverflow are less connected. Linking these accounts is a prerequisite of behavior mining across these two communities. In this section, we define the identity linkage problem, and then give our method and its experimental results.

3.1. *Problem definition*

Let P denote the collection of all natural persons, U_g denote the collection of users in GitHub and U_s denote the collection of users in StackOverflow. Let $T(u) = p$ be a mapping function to map a user in GitHub or StackOverflow to a nature people in the real world. Now our goal is to find such a function f to solve the identity problem. For a user pair (c_1, c_2), where $c_1 \in U_g$ and $c_2 \in U_s$, $f(c_1, c_2) \to \{0, 1\}$. If $T(c_1) = T(c_2)$, which means c_1 and c_2 refer to the same nature people, then $f(c_1, c_2)$ equals 1, else $f(c_1, c_2)$ equals 0.

We want to use the machine learning method to train such a model f. The key issue is how to choose some efficient features for the model. Let U_s denote the set of user pairs combined by the users among GitHub and StackOverflow. These features should be able to differentiate the user in the user pair which makes the value of function f equal to 0 and be similar for users in the user pair which makes the value of function f equal to 1.

3.2. *Feature extraction*

To solve the identity linkage problem defined above, we extract three kinds of features to calculate the similarities between two users, which are username features, user behavior features and user writing style features.

3.2.1. *Username features*

Since people usually use similar usernames in different communities, it is very significant to extract useful features from usernames. Four string matching algorithms are chosen to measure the username similarity: (1) Levenshtein distance, which is the number of transfers needed to transfer one string to another one; (2) Jaro–Winkler distance, which is commonly used for measuring the similarity of short strings especially for usernames; (3) longest common substring, which is the longest string that is a substring of two strings; (4) longest common subsequence, which is the longest subsequence common to two strings.

The value of the Jaro–Winkler distance is in the range [0,1], and we compute the ratio and transfer the values of the other three methods in the same range. In the experiments, we will set all these metrics as features to predict the identity linkage using the decision tree model, and two of them with the best performance will be chosen.

3.2.2. *User behavior features*

Existing empirical studies about social behaviors (see e.g. [8]) show that a user's social behavior exhibits a surprisingly high level of consistency across different communities over a sufficiently long period of time. It is rational to hypothesize that two users in GitHub and StackOverflow correspond to the same nature people in real world if they have a high level of synchrony.

For each repository in GitHub or each question in StackOverflow, there is a tag labeled to describe the programming language or related technologies. Therefore, we can obtain topics in user behaviors with these tags, and measure the similarity of two user behaviors by the distribution similarity of the topics in their behaviors.

However, the tagging systems in these communities are very different. For example, the numbers of tags in GitHub and StackOverflow are 57 and 21,300, respectively. Tags are marked by the system automatically in GitHub, but by users themselves in StackOverflow. To solve this problem, we converse those synonymous tags into the same tags based on a synonyms relation,[c] and a common set of all tags both in GitHub and StackOverflow are extracted. Then these tags are used to build the distribution of topics in user behaviors.

If a high level of synchrony is observed over an extended period of time between two user accounts from different platforms, it is reasonable to hypothesize that these two users correspond to the same person.

Another problem we encountered is that people are not always using different communities at the same time so that the amount of information might be missed in such a process. Therefore we propose a user behavior matching method inspired by bio-stimulation [9] to reduce the impact of that problem. The main idea of the bio-stimulation is that the maximum stimulation from a pooled signal set plays a

[c] http://stackoverflow.com/tags/synonyms.

significant role for perception. So we segment the user behaviors by different time periods. For example, we divide developers' behaviors into four quarters for each year and three months for each quarter.

Suppose each user pair (u^1, u^2) where $u^1 \in$ GitHub and $u^2 \in$ StackOverflow. For each month, we obtain all the language tags of the projects u^1 in GitHub and get a tag distribution (TD) for the tags belonging to the common set. At the same time, we also catch all the questions u^2 asked on the StackOverflow in this month and get a tag distribution. Then a cosine similarity is used to measure the user's behavior similarity for the month, denoted as $s_{mr}(i)$. Following formula is defined to calculate the similarity of user behaviors for a quarter and for a year, where N is the number of months. In our experiment, we measure an average similarity of user behaviors per year as the final behavior similarity between two users:

$$S_{mr} = \frac{1}{N}\left(\sum_{i=1}^{N}(s_{mr}(i))^q\right)^{\frac{1}{q}}, \quad q \geq 1. \tag{1}$$

3.2.3. *Writing style features*

As mentioned above, unlike the traditional social networking, all user-related data on the GitHub and StackOverflow are textual, which makes the available features very restrictive. So in the identity linkage across the two communities, how to extract more valid features in the available data is still a problem.

Most of user-generated data in the GitHub and StackOverflow are textual. Some studies [10–12] have shown that the user's writing style can help to achieve reliable results in user recognition situations. To calculate the user's behavior similarity for a quarter, the $s_{mr}(i)$ is the user's behavior for each month in the quarter, and N is 3. Similarly, we calculate the user's behavior similarity for a year based on the similarity for the quarter. Finally, we calculate an average similarity of the user based on each year and obtain a final similarity among two users. So we apply the method in [12] to extract user's writing style features, listed in Table 1. Then, the writing style similarity between two users is measured by KL-divergence, using those features.

Table 1. Writing style features.

Feature	Definition
Length	Number of different words
Vocabulary richness	Frequencies of hapax legomena and dis legomena
Word shape	Frequencies of words with different combinations of upper and lower case letters
Word length	Frequencies of words that have 1–20 characters
Letters	Frequencies of a–z, ignoring case
Digits	Frequencies of 0–9
Punctuations	Frequencies of . ? ! , ; " ()
Function words	Frequencies of words like "the", "of" and "then"

3.3. *Identity linkage*

With all the features above, we train an identity linkage model on a ground-truth dataset, using CART. By this model, we can obtain the probability of whether the user pair refers to the same user. And then, the identity linkage problem is converted to a matching problem in bigraph. Suppose user u^1 in GitHub and each candidate user u^2 in StackOverflow, a conditional Heuristic Greedy Matching (HGM) is used to avoid false positive matching and it finally generates a candidate user pair as (u^1, u^2).

For example, in Fig. 2, each user pair has a probability which is assigned by the CART decision tree. In HGM, as shown in Fig. 2(a), the information generated by each user is calculated firstly. Suppose user X is the owner of the most information and his best candidate is X'. User pair (X, X') will be the first linkage selected by HGM because (X, X') shares the highest probability. After selecting, X and X' will be deleted in the candidate list before our next matching. Finally, three user pairs (X, X'), (Y, Y') and (Z, Z') are matched as shown in Fig. 2(b).

3.4. *Experiments*

The data we use is all before May 2016 from ghtorrent (see Footnote a) and archive.org (see Footnote b). To validate our approach, the first thing is to construct ground-truth data. In previous papers, authors often use email data as a criterion for determining whether it is the same user in different communities. However, many communities, including StackOverflow, began to pay more attention to user privacy information protection. From 2016 onwards, StackOverflow no longer directly provides the user's mailbox information. We know that a few users provide their profile URLs, so we consider that if the profile URL of a user in StackOverflow and that of a user in GitHub both link to the same site, then they are the same person.

However, we cannot simply link users by this way as in our statistics there are less than 10% users providing their profile URLs. In these 10% users, we totally link 16,000 users, corresponding to 8000 people, by profile URLs. Therefore, we use these

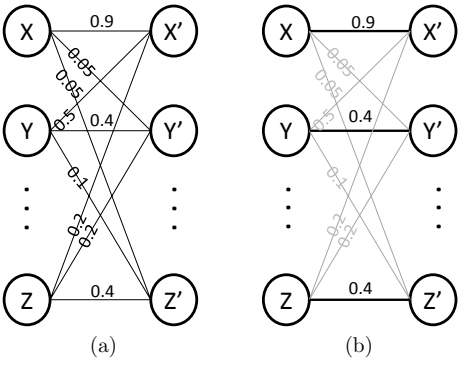

Fig. 2. An example of HGM algorithm.

8000 people, who are divided into five groups, to adopt five-folder cross-validation to evaluate the performance of our method. There are 405,305 questions and answers posted in StackOverflow and 321,342 repositories created or attended by these 8000 users.

3.4.1. *Selection of username metrics*

With the decision tree model, all potential combinations from four metrics are set as features to do prediction, and the best combination is selected. The result is illustrated in Fig. 3, where Levenshtein distance, longest common substring, longest common subsequence and Jaro–Winkler distance are abbreviated by 1, 2, 3 and 4, respectively. The experiment results show that the combination of Levenshtein distance and longest common subsequence has the best performance.

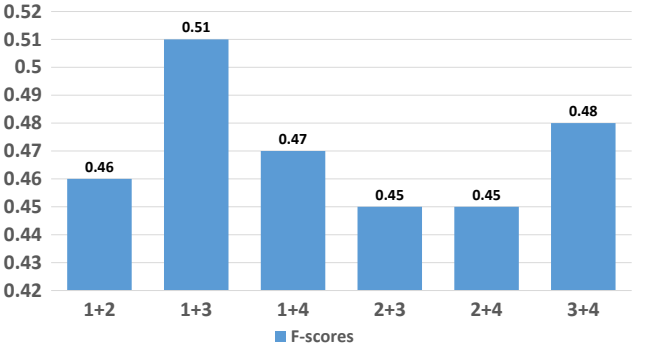

Fig. 3. Results for combination of different username metrics.

3.4.2. *Feature contribution*

In this experiment, all kinds of features above are used to train the decision tree model. They are username similarity (abbreviated by U), user behavior similarity (abbreviated by B) and user writing style similarity (abbreviated by W). Figure 4 illustrates how the model is affected by each feature and the combination of features. It is demonstrated that the model trained with all features has the best performance.

3.4.3. *Comparison with other methods*

In this experiment, we compare our method with other two methods, the TBIL [13], and the Bird *et al.*'s method [4], to solve the identity linkage problem across software communities. In Bird *et al.*'s method [4], it pays attention to the username and email of the user. However, as we mentioned above, not all users will use the same or similar usernames among different software communities, moreover, some users are not willing to offer their email address to the public. For such users, it is hard to use this

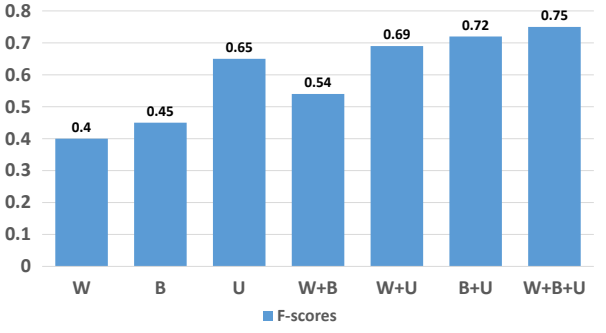

Fig. 4. Contribution of different features.

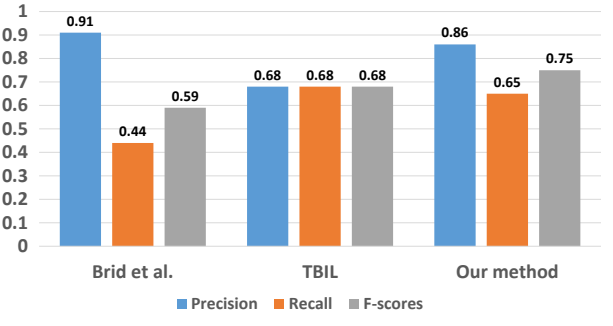

Fig. 5. Comparison with other methods.

information to link identities. The TBIL is a state-of-the-art method, which uses the information of username, user topic and user skill.

Figure 5 illustrates the results of the comparison. Bird *et al.*'s [4] method achieves a high precision (around 0.91), but low recall and F-score. That is because not all users use similar usernames among different communities and some users do not offer their email addresses. Since the TBIL uses a full matching strategy, it achieves a same precision, recall and F-score (0.718). Our method has a good precision and recall, and gets the best F-score (around 0.75). Compared to those methods, our method adopts a bio-stimulation method with more features, which can analyze the user's topics and actions meticulously, and thus has a better performance. Furthermore, we use a conditional greedy-based matching method to avoid false positive problem, since not all users have accounts in both GitHub and StackOverflow.

4. Mining Developer Behavior Across GitHub and StackOverflow

In this paper, we want to explore the behaviors of developers in the open-source community (e.g. GitHub) and the knowledge-sharing community (e.g. StackOverflow), to discover the interrelationships among the various types of information, and

to explore if different types of developers will have significantly different behaviors. In addition, we expect to get insight into knowledge sharing in the software development process, for example, are the developers' participation in the sharing of knowledge affected by time, project schedule, personnel types or other factors? By mining the behavior of developers in the cross-platform community, we can have more in-depth understanding of the habits and hobbies of developers so that we can help developers to complete the task and improve development efficiency better.

4.1. *Research questions*

Three main research questions are designed as follows:

> **RQ1.** Do one developer's activities in GitHub reflect his activities in StackOverflow? If yes, how?

We wonder if the developer's behaviors in GitHub and StackOverflow are relevant, and if the productivity of GitHub developer is relevant to his participation in StackOverflow. For example, are active issue committers in GitHub also active question askers in StackOverflow? Do more users ask for more issues on GitHub than answers on StackOverflow? Do users with fewer repositories have less behavior on StackOverflow? We use the statistical test method such as T-graph to draw the initial sketch and the scatter and fitting curve.

> **RQ2.** How is the relevance of developers' concerned topics between GitHub and StackOverflow?

By our experience in software development, if a developer who participates in the development of a Java project encounters a lot of Java errors or problems, then he is very likely to be concerned on Java-related contents in knowledge-sharing community. Therefore, we attempt to know how is the relevance of developers' concerned topics between GitHub and StackOverflow and if the developers' concerns in StackOverflow will shift over the time of their current participating projects in GitHub. By collecting the developers' textual contents in GitHub and StackOverflow, we will measure the similarity between their topics.

> **RQ3.** Which kind of activities in StackOverflow are more relevant to the developer's concerns in the software development process?

We will also explore which one will be closer to developers' concerned topics in GitHub among their questions, answers and comments in StackOverflow, and which

activity in StackOverflow is more representative of what developers concern in the software development process. We introduced how to use our method to identify more identities of the same identity developers and eventually got 40,000 users of the same. We extracted out various types of data of the 40,000 developers, including name, age, repository description (or readme file), issue data in GitHub and question, answer and comment data in StackOverflow.

4.2. *Behavior mining*

Guided by the above research questions, we collect and extract the following developer behavior data from GitHub and StackOverflow: the numbers of developers' repositories in GitHub; the numbers of developers' questions, answers, comments and age in StackOverflow; descriptions (or readme file) of repositories, and issue data in GitHub; and textual contents of questions, answers and comments in StackOverflow.

4.2.1. *T-graph analysis*

We summarize the results of T [14], a multiple contrast test procedure using 5% familywise error rate, by means of T-graphs [15] shown in Fig. 7. For Tukey-type contrasts, we summarize the results of T by means of T-graphs. In such a directed acyclic graph, nodes correspond to the different groups being compared, and edges to the results of the pairwise comparisons. There is an edge from A to B if A tends to have higher values for a given metric than B (i.e. for the comparison A–B, T reports $p < 0.05$). Since T respects transitivity, in a T-graph we omit direct edges between A and B if there is a path from A to B passing through at least one other node. Consider the example T-graph in Fig. 7, summarizing the results of the T-procedure applied to four groups of values A, B, C and D: D tends to have higher values than both B and C, but lower than A; A tends to have higher values than all other groups (D directly, B and C transitively).

4.2.2. *LDA-based topic clustering*

We know that developers will ask questions, make answers and comments on others at StackOverflow. At the same time, the developer will leave a lot of text information on GitHub. We want to know what content the developers are concerned about in these two platforms. In addition, we want to know which is the most representative content in the process of communication between developers and which activities (question/answer/comment) are closer to the content of the software development process.

We follow a state-of-the-art algorithm, Latent Dirichlet Allocation (LDA), to obtain the topic distribution for each user. LDA has been shown to be effective to process various text data in many domains, such as software engineering and social

network analysis. The inputs of LDA algorithm are a set of documents and the number of topics K, and the outputs are the topic distribution of each input document. In our problem, some users may generate not only one document, we thus merge all documents generated by a user to D to further apply LDA. Because LDA is a kind of bag-of-words model, before applying it, each document D must be converted to a bag of-words vector. Firstly, we remove all stopwords which are used in almost every document such as the, is, etc. Then we use stemming to reduce words to their root form. Finally, we filter words that only appear in one community and words appearing in a low frequency (appear less than five times in all documents). After obtaining a text vector for each user, we set text vectors of all users from both communities as input and then use LDA to get the topic distribution for each user.

On the other hand, to decide the best number of topics K, we use a simple approach proposed in [16], which states that the best number of topics is the one with the highest log likelihood value. Therefore, we build LDA model several times with different K, and then choose the K with the highest log likelihood value. After obtaining the topic distribution for each user, we use the symmetrical KL-divergence and cosine similarity to calculate the similarity of topic distributions.

4.2.3. *Cross-site tagging*

Documents in StackOverflow are labeled with tags, such as questions are labeled by their owners. These tags are all terms of software engineering, and so can be treated as user skill descriptions. If only one community has the tagging system, we can also use the cross-site tagging method to build a model based on labeled data in this community and then use this model to label the other community.

The tagging systems in StackOverflow and GitHub are very different. According to the principles set forth in [13], we can mark GitHub with tags in StackOverflow using cross-site tagging method which consists of two steps: (1) Unnecessary tags removal in StackOverflow: Interestingly, we found that 20% of tags could cover all questions in StackOverflow by experiments. Therefore, we remove some low-frequency tags and rewrite the remaining 80% tags by some rules [13]. (2) Tag transfer from StackOverflow to GitHub: Since all questions and answers in StackOverflow (every answer is linked to a tagged question) are marked, these tagged data are used to train a naive Bayes model for text classification. Then each repository in GitHub is labeled and gets a TD using this model, based on the text contents in the readme file or project instruction.

To get the final tag distribution, we apply a technique named spreading activation to infer associated tags. During the process of spreading activation, for each tag in TD, in one iteration (Fig. 6), it propagates its weight to other tags with corresponding similarity. Lastly, we use the symmetrical KL-divergence and cosine similarity to calculate the similarity of tag distributions.

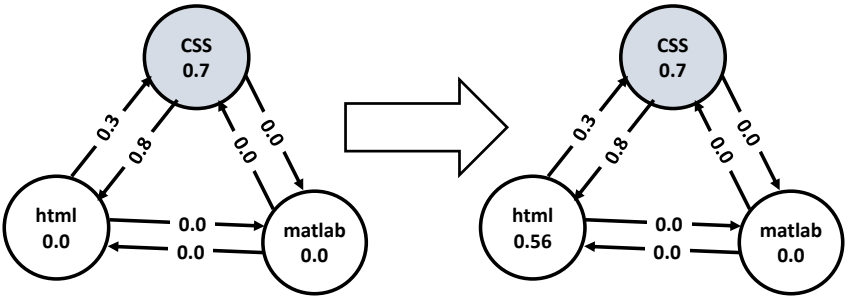

Fig. 6. An example of spreading activation in one iteration.

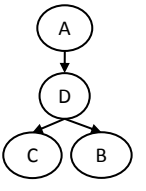

Fig. 7. The T-graph.

4.3. *Experiments and findings*

We conduct several experiments by the above technologies to answer three research questions.

For RQ1, we observe the distributions of the numbers of issues and questions. For each ordered pair of issues and questions, with the data sorted along one dimension, we split the other dimension into many groups and compare the distributions.

We divide the data into four pieces firstly and then adopt the T-graph method to analyze. Figure 8 shows that the most and second active 25% of the issue committers (Q1 and Q2) ask more questions in StackOverflow than other quartiles (Q3 and Q4), but Q1 and Q2 cannot be distinguished. This phenomenon is consistent with the result using polynomial fitting as shown in Fig. 9 that active issue committers are also active question askers.

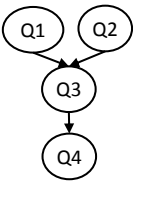

Fig. 8. Q1 and Q2 ask more questions on StackOverflow than others.

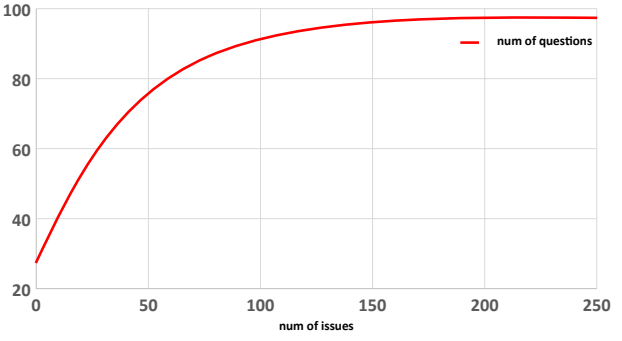

Fig. 9. Relationship between the number of questions and the number of issues.

Active issue committers are also active question askers.

For RQ2, we measure everyone's similarity of his textual contents between GitHub and StackOverflow by LDA and the tagging system. The similarities of developers are sorted from low to high, and illustrated by a fitting curve in Fig. 10. It is easy to observe in the figure that a small amount of developers' value is very low (only 0–0.28), but large number of developers have the similarity value from 0.3 to 0.5. The similarity between the contents of developers' concerns in GitHub and StackOverflow is about 0.45. In addition, the experimental result shows that the cross-site tagging system outperforms the LDA method, because those tags maintained by StackOverflow already have a high degree of summary of the textual content.

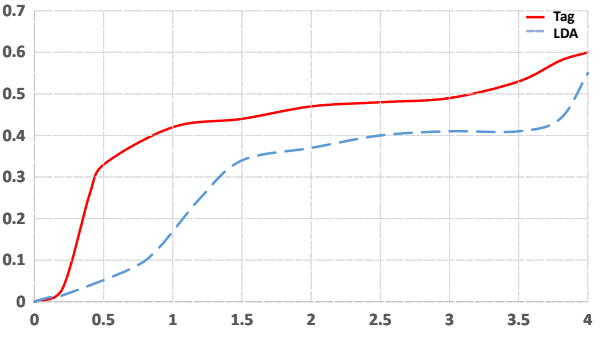

Fig. 10. The similarities measured by cross-site tagging system and LDA.

For most developers, the topics of their contents in GitHub are similar to those of their questions and answers in StackOverflow.

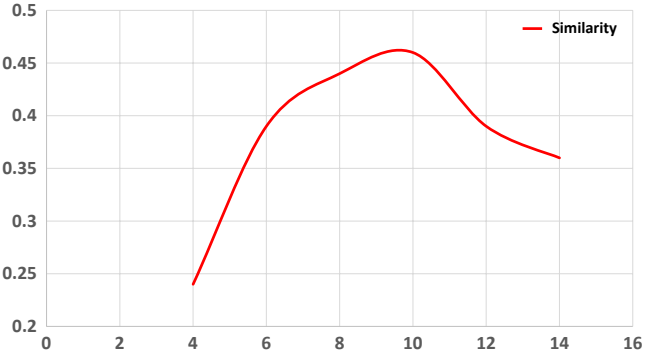

Fig. 11. The similarity variation with the time of projects.

And then, we conduct an in-depth research on RQ2. Suppose a developer participates in the project R, the readme file or description text of R is set to D_g and the developer's direct participation in the project R starts at time T_s. Setting the time interval as Δt months, we obtain all questions, answers and comments of this developer in StackOverflow in $T_s \pm \Delta t$ and set to D_s. Then, the tagging system helps us to get the tag distributions for all D_s and D_g, and the KL-divergence is used to calculate the similarity between D_s and D_g. In this experiment, we set Δt to 4, 6, 8, 10, 12 and 14. And for each Δt, we simply filter out the upper and lower 10% extremums of the calculated similarities. The median value of the remaining similarity data is taken as the ordinate, and the abscissa is Δt as shown in Fig. 11. When Δt is 8 or 10, the contents that developers are concerned about in GitHub and StackOverflow have the highest correlation, and the similarity is low when Δt is less than 8 or greater than 10. So, we speculate that developers will pay more attention to some of the project-related areas in a period of time.

> Developers' concerns in StackOverflow shift over the time of their current participating projects in GitHub.

For RQ3, we respectively compare the questions, answers and comments of each developer in StackOverflow to the textual contents they left in GitHub. Figure 12 shows that the textual contents that developers left in GitHub are more relevant to their answers than questions and comments in StackOverflow. In other words, the developer's answers are more representative of developer's concern in the software development process.

> The contents of developers' concerns in GitHub are more relevant to their answers than questions or comments in StackOverflow.

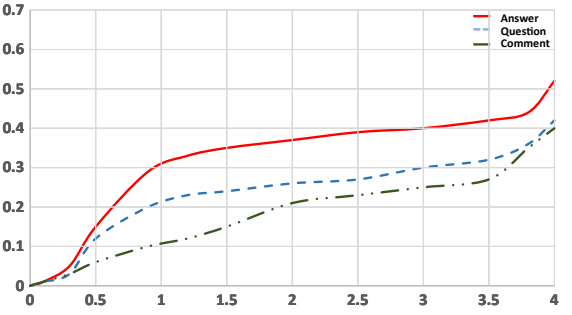

Fig. 12. Comparing different activities in StackOverflow to the Contents in GitHub.

5. Conclusion and Future Work

In this paper, we propose a novel approach for developer behavior mining across GitHub and StackOverflow. It links the accounts from GitHub and StackOverflow, by leveraging the features from usernames, user behaviors and writing styles. Then, it mines the developer behavior data across these two communities, and gains some valuable findings. In the future, we plan to research on the identity linkage problem among more than two software communities and continue to explore how StackOverflow influences GitHub, for example, what kind of issues or problems always be posted to StackOverflow and what are the performances of software developers with different programming abilities in these two communities.

Acknowledgments

This research is supported by National Natural Science Foundation of China (Grant No. 61472242) and 973 Program in China (Grant No. 2015CB352203).

References

1. W. N. Robinson, T. Deng and Z. Qi, Developer behavior and sentiment from data mining open source repositories, in *Proc. 49th Hawaii Int. Conf. System Sciences*, 2016, pp. 3729–3738.
2. E. Kouters, B. Vasilescu, A. Serebrenik and M. G. van den Brand, Who's who in GNOME: Using LSA to merge software repository identities, in *Proc. 28th IEEE Int. Conf. Software Maintenance*, 2012, pp. 592–595.
3. C. Treude, O. Barzilay and M.-A. Storey, How do programmers ask and answer questions on the web? (NIER track), in *Proc. 33rd Int. Conf. Software Engineering*, 2011, pp. 804–807.
4. C. Bird, A. Gourley, P. Devanbu, M. Gertz and A. Swaminathan, Mining email social networks, in *Proc. Int. Workshop Mining Software Repositories*, 2006, pp. 137–143.
5. M. Goeminne and T. Mens, A comparison of identity merge algorithms for software repositories, *Sci. Comput. Program.* **78**(8) (2013) 971–986.

6. T. Ahmed and A. Srivastava, Understanding and evaluating the behavior of technical users: A study of developer interaction at StackOverflow, *Hum.-Centric Comput. Inf. Sci.* **7**(1) (2017) 8.
7. B. Vasilescu, V. Filkov and A. Serebrenik, StackOverflow and GitHub: Associations between software development and crowdsourced knowledge, in *Proc. Int. Conf. Social Computing*, 2013, pp. 188–195.
8. G. Pickard, W. Pan, I. Rahwan, M. Cebrian, R. Crane, A. Madan and A. Pentland, Time-critical social mobilization, *Science* **334**(6055) (2011) 509–512.
9. S. Liu, S. Wang, F. Zhu, J. Zhang and R. Krishnan, Hydra: Large-scale social identity linkage via heterogeneous behavior modeling, in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2014, pp. 51–62.
10. R. Zheng, J. Li, H. Chen and Z. Huang, A framework for authorship identification of online messages: Writing-style features and classification techniques, *J. Am. Soc. Inf. Sci. Technol.* **57**(3) (2006) 378–393.
11. A. Abbasi and H. Chen, Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace, *ACM Trans. Inf. Syst.* **26**(2) (2008) 7.
12. A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin and D. Song, On the feasibility of internet-scale author identification, in *Proc. IEEE Symp. Security and Privacy (SP)* (IEEE, 2012), pp. 300–314.
13. W. Mo, B. Shen, Y. Chen and J. Zhu, TBIL: A tagging-based approach to identity linkage across software communities, in *Proc. Asia-Pacific Software Engineering Conf.*, 2015, pp. 56–63.
14. F. Konietschke *et al.*, Rank-based multiple test procedures and simultaneous confidence intervals, *Electron. J. Stat.* **6** (2012) 738–759.
15. B. Vasilescu, A. Serebrenik, M. Goeminne and T. Mens, On the variation and specialisation of workload: A case study of the GNOME ecosystem community, *Empir. Softw. Eng.* **19**(4) (2014) 955–1008.
16. T. L. Griffiths and M. Steyvers, Finding scientific topics, *Proc. Natl. Acad. Sci. USA* **101**(Suppl. 1) (2004) 5228–5235.