Soft Computing (2020) 24:5683–5693
https://doi.org/10.1007/s00500-019-04217-7

FOCUS

Core-reviewer recommendation based on Pull Request topic model and collaborator social network

Zhifang Liao[1] · ZeXuan Wu[1] · Yanbing Li[1] · Yan Zhang[2] · Xiaoping Fan[3] · Jinsong Wu[4]

Published online: 15 July 2019

Abstract
Pull Request (PR) is a major contributor to external developers of open-source projects in GitHub. PR reviewing is an important part of open-source software developments to ensure the quality of project. Recommending suitable candidates of reviewer to the new PRs will make the PR reviewing more efficient. However, there is not a mechanism of automatic reviewer recommendation for PR in GitHub. In this paper, we propose an automatic core-reviewer recommendation approach, which combines PR topic model with collaborators in the social network. First PR topics will be extracted from PRs by the latent Dirichlet allocation, and then the collaborator–PR network will be constructed with the connection between collaborators and PRs, and the influence of each collaborator will be calculated via the improved PageRank algorithm which combines with HITS. Finally, the relationship between topics and collaborators will also be built by the history of PR reviewing. When a new PR presents, a collaborator will be chosen as a core reviewer according to the influence of collaborators and the relationship between the new PR and collaborators. The experiment results show in the matching score calculation processing, the influence of collaborators shows higher than that with the expert, and the recommendation precision is better than 70%.

Keywords Pull Request · Core-reviewer recommendation · GitHub · LDA · Social network

1 Introduction

GitHub, a popular open-source community (Begel et al. 2013; Liao et al. 2018), has attracted the participations of tens of millions of developers and millions of open-source projects. Pull Request (PR) is a major contributor to external developers of open-source projects (Gousios et al. 2014). When an external developer has implemented some new features or fixed some bugs, he or she can contribute his or her code via submitting a PR. Then, the core developers decide how to deal with the PR after taking all the opinions of reviewers into consideration.

With the rise of big data (Kuang et al. 2018) and service computing (Liao et al. 2019; Li et al. 2018), GitHub is becoming more and more hot and popular. The recent studies showed that the popular projects receive nearly 100 PRs per day from external contributors (Liao et al. 2018). To improve the efficiency of PR reviewing, some researchers (Yu et al. 2014; Balachandran 2013; Thongtanunam et al. 2014) have proposed some proposals to recommend appropriate reviewers for new PR. However, the reviewers they recommend include any developers, whatever the core developer or external developer he or she is. If the recommender is an external developer, he or she just can review the PR, but he or she cannot decide to refuse or merge the PR. The PR still needs waiting for a core developer to do the final decision. The delay might be

Communicated by B. B. Gupta.

✉ Xiaoping Fan
xpfan@csu.edu.cn

Zhifang Liao
zfliao@csu.edu.cn

1 School of computer Science & Engineering, Central South University, Changsha, China

2 Department of Computer, Communication and Interactive System, School of Engineering and Built Environment, Glasgow Caledonian University, Glasgow, UK

3 Hunan University of Finance and Economics, Changsha, China

4 Department of Electrical Engineering, Universidad de Chile, Santiago, Chile

a long time. However, if the recommender is the core developer, the delay can be reduced greatly.

In this paper, we propose a NTCRA (Social Network and Topic Model-based Core-Reviewer Recommendation Algorithm) algorithm to match the appropriate collaborators as the reviewer for new PR. However, the active reviewers are always recommended frequently in the majority reviewer recommendation methods. To reduce the recommendation frequency of the active collaborators, we calculate the influence of each collaborator by the collaborator–PR network. Then, we use the influence as the weight to calculate the recommendation score between each collaborator and the new PR.

The structure of this paper is organized as follows. We present the existing research results in Sect. 2. Section 3 introduces the algorithm methodology and details of NTCRA. We explain the results of experiment and its validation in Sects. 4 and 5. Section 6 introduces the possible problems and draws conclusions.

2 Related work

2.1 Influential people mining

Influential people mining is a popular issue in the social network. Currently, most of the influence calculation methods are based on PageRank, HITS, and their methods of variation. Hassan Sayyadi (2009) constructed an author-paper heterogeneous network, and proposed a method to rank the nodes which combines HITS and PageRank. Hassan Sayyadi used HITS to calculate the importance of paper, and used the PageRank to calculate the importance of author. Thung et al. (2013) proposed a network of developers and projects, and applied the PageRank algorithm to calculate the weight of the nodes. Fan et al. (2018) proposed a kind of discovery algorithm based on local core members to solve the problem of community detection in social networks, and get the importance of each node in the process. Yang et al. (2012) studied the graph structure and random walk model and proposed the SocialRank algorithm to calculate the individual influence. Li et al. (2015) proposed a measure called CommRank, which calculates the influence of communities in the social network. And based on the CommRank, they improved algorithm to deal with the problem of maximizing influence. We means that most of the methods described above are based on HITS, PageRank or Katz models. These methods have better results than traditional methods based on degree and closeness.

2.2 Reviewer recommendation

As an essential part of GitHub, more and more researchers have focused on improving the efficiency of PR reviewing. Zhang et al. (2014a, b) conducted an exploratory study of @-mention in PR-based software development, and found that @-mention is beneficial to speed up the review of PR. However, due to the large number of people in the open-source community, developers are not able to find the suitable reviewers quickly and accurately. To solve this problem, some researchers have proposed some reviewer recommenders. For example, Balachandran et al. (2013) developed a Review Bot to find developers who frequently submit code changes as reviewers. Later, Thongtanunam et al. (2014) proposed a method of using the file path to find reviewers. Yu et al. (2014) proposed a reviewer recommendation algorithm based on the review network. Lipcak et al. (2018) conducted big data experiments on various methods above and found that the results of these methods for large-scale projects are not very satisfactory, but they have a good effect on medium-scale projects. Xia et al. (2017) proposed a recommendation algorithm that considers implicit relations and neighborhood models. This method can extract the possible implicit information in the PR comment records, and then are collided and filtered through the neighbor model to obtain the final recommender. Yang et al. (2018) proposed a two-layer reviewer recommendation model that matches by combining recommendation scores with reviewer types. This method sets up different types of reviewers based on the common recommendation methods, and distinguishes and recommends the reviewers from the perspective of technology and management.

All these recommendation methods include text analysis, deep learning (Chang et al. 2017; Gong et al. 2017; Li et al. 2017), collaborative filtering, and multiple network (Deng et al. 2019), but those existing methods are lack of considerations of the multi-dimensional features (Zhang et al. 2018) of prediction (Kuang et al. 2018, 2019) and recommendation. Meanwhile, their approaches focus on recommending reviewers for new PRs, but the recommended reviewers include all developers, not only the core developers. In open-source ecosystem like GitHub (Liao et al. 2018, 2019a, b), if the recommended reviewer is not a core developer, the reviewer cannot merge or refuse the PR when he or she thinks the PR meets or does not meet the demand of the project directly. The PR still needs waiting core developer to do the merging or refusing operation. However, if the reviewer is a core developer, he or she can merge or refuse the PR directly in that case. And all these approaches face the repeating recommendation problems of active reviewers which will bring a heavy workload to the active reviewer. A large amount of research content above indicate that there is still a huge room for improvement of PR review.

3 Methodology

In GitHub, the core developers are the managers of the project. They can submit the revisions to the main repository directly, and they can refuse or merge the PR which the external developers submit. These core developers are called collaborators of the project in GitHub. Usually, a collaborator gets higher expertise of the project than the external developers. Hence, if they are recommended as the reviewers to PRs, he or she would make a decision to merge/refuse the PR quickly. In this case, the collaborators are matched as a reviewer of that PR. As the rule, a PR just can be merged or refused by the collaborators of the project. In other words, if a reviewer is not a collaborator, he/she will not have the permission to merge or refuse the PRs. If the reviewer is a collaborator, he/she can make the final decision immediately after reviewing the PRs.

In order to better introduce our methods and related nouns, we give the following definitions.

Definition If the reviewer is a collaborator, he/she will be called core reviewer of the PR.

Usually, the title and description of one PR reflect the theme of the PR. We propose a NTCRA algorithm to calculate the relationship between the theme and reviewer (only include collaborators). Generally, a higher influence a person gets, the more possibility he will be a collaborator. That means, he will get more work to do and sometimes the situation will decrease the PR integrating efficiency. To reduce the frequency of active collaborator recommendation, we construct a collaborator–PR heterogeneous network in NTCRA to calculate the influence of each collaborator. In NTCRA, if a developer reviews that PR, we assume that he or she is interested in the theme of that PR. The topics which represent the theme of the PR will be extracted from PRs via LDA model (Blei et al. 2003; Kuang et al. 2018). And the topic-document distributions will be used to calculate the relationship between topics and collaborators. Meanwhile, to reduce the cost of topic-document distributions calculation of new PRs, we use the distributions calculation of new PRs, and obtain the topic distributions of new PR from the word-topic distributions which generated by topic extracted processing previously. According to the influence of collaborator calculated by collaborator–PR network, we can work out at the matching scores for each collaborator from the relationship between topics and collaborators. The collaborator who gets the highest score will be recommended as core reviewer to the PR. The overview of the proposed algorithm NTCRA is shown in Fig. 1. We will expand more details of the NTCRA algorithm in the following sections.

3.1 Collaborator–PR network construction

We construct the *Collaborator–PR Network* individually in every project. In a given project, the reviewing relationship between PRs and collaborators is many to many. As shown in Fig. 2, there are many Pull Requests in Project *P*. A collaborator can review several PRs, and a PR can be reviewed by several collaborators more than once (posting one comment represents reviewing once). For example, *Collaborator* C_1 reviewed *Pull Request* PR_1 W_1 times, and reviewed *Pull Request* PR_2 W_2 times. *Pull Request* PR_2 is reviewed by C_1, C_2, C_3, respectively.

The *Collaborator–PR network* is defined as a heterogeneous network, and it is a weighted undirected graph which includes two types of nodes and two types of edges. The two types of nodes are collaborators and PRs, the two types of edges are review edges and common interest edges, shown as in Fig. 3. If *collaborator* c_i reviewed PR_j at least once, there is an edge w_{ij} between c_i and PR_j. The weight w_{ij} is defined as the reviewing times of PR_j. If collaborator c_i and c_j reviewed a PR together, there is a common interest shared between them. Hence, there is an edge between collaborator c_i and c_j, and the weight is defined as the number of PRs reviewed by them together. (All of them have posted comments to the PR.)

3.2 Collaborator influence calculation

In order to better analyze the corresponding rules of the influence network, we partition the collaborator–PR network into two networks. The network on the left contains collaborators and PRs, and they are connected with each other through review edges. This network is a bipartite network which can map onto the HITS network; we define collaborators as hub nodes and PRs as authority nodes. Thus, we can use the HITS to transfer authority scores between collaborators and PRs. The network on the right contains the collaborator nodes and common interest edges, so we can use PageRank algorithm to calculate authority scores between collaborators. Figure 4 shows the example of partition.

Traditional approaches consider the network structure as non-weighted; the propagation process is in uniform distribution. In fact, the relationship between different collaborators and PRs is different. Hence, the propagation should be different too. In this paper, we propose an asymmetric strategy to pass authority scores. We use propagated matrices to show the propagated process; the propagation matrices are calculated as follows:

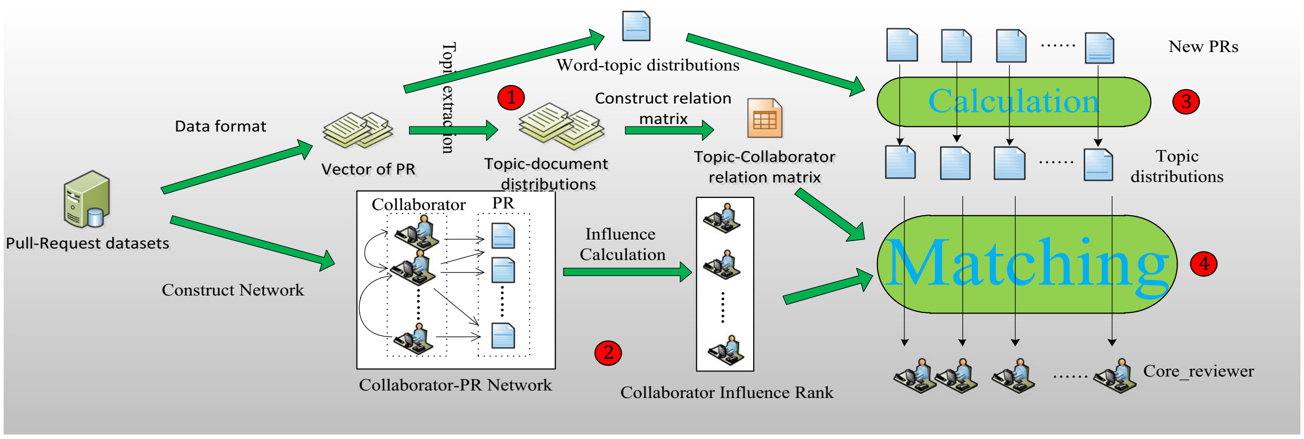


Fig. 1 Overview of the algorithm NTCRA. NTCRA includes four steps: (1) generates the word-topic and topic-document distributions, and builds the relation matrix between collaborators and topics via the topic-document distributions; (2) builds the collaborator–PR network and calculates the influence of each collaborator; (3) calculates the topic-document distributions by using the word-topic distribution generated in previous topic generation process; (4) calculates the matching scores according to the influence of collaborators and the relation matrix

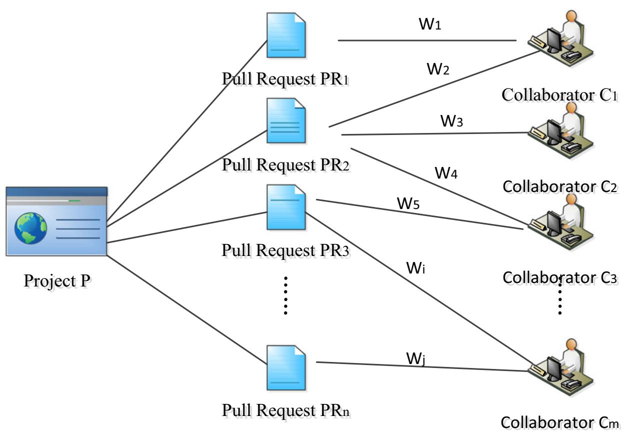


Fig. 2 Review relations between PRs and collaborators

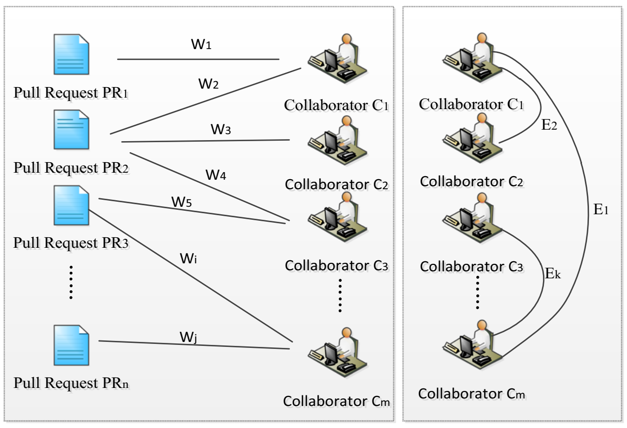


Fig. 4 Network Decomposition: a review network and an interest network

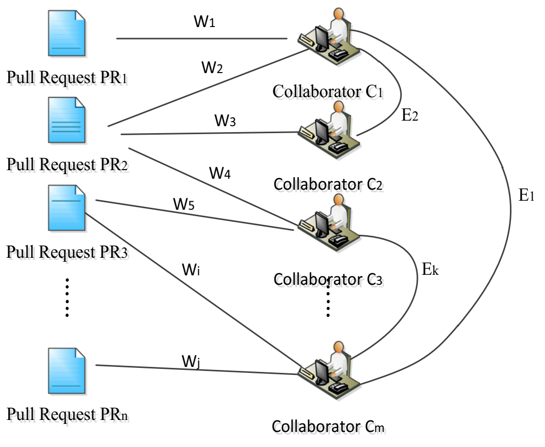


Fig. 3 Example of *collaborator–PR network*

$$Mcp(i,j) = \frac{Mr(i,j) \times w_{ij}}{\sum_{i=1}^{|C|} w_{ij}} \tag{1}$$

$$\mathrm{Mpc}(i,j) = \frac{Mr(i,j)^T \times w_{ji}}{\sum_{j=1}^{|PR|} w_{ij}} \tag{2}$$

$$\mathrm{Mcc}(i,j) = \frac{Mc(i,j) \times e_{ij}}{\sum_{i=1}^{|C|} e_{ij}} \tag{3}$$

where $Mcp(i,j)$ is the propagated matrix from collaborators to PRs; Mpc(i,j) indicates the propagated matrix from PRs to collaborators; $Mcc(i,j)$ represents the propagation matrix from collaborators to collaborators; $Mr(i,j)$ is the adjacency matrices of review network; if collaborator i reviewed PR j, then $Mr(i,j) = 1$, otherwise $Mr(i,j) = 0$; $Mc(i,j)$ is the adjacency matrices of interest network; if collaborator i, j reviewed a PR together at least once, then $Mr(i,j) = 1$, otherwise $Mr(i,j) = 0$; w_{ij} is the weight in review network; e_{ij} is the weight in interest network.

We use vectors to represent the ranking scores, where $R(C)$ represents the ranking scores of collaborators and $R(\mathrm{PR})$ represents the ranking scores of PRs. We use 1/n to initial the PR ranking scores, and use 1/m to initiate the collaborator ranking score, where n and m correspond to the number of PRs and collaborators, respectively. We update the ranking scores by the iteration steps, until the error between two iterations less than the error value set previously. The iteration steps are defined as follows:

$$R_k(\mathrm{PR}) = Mcp \times R_{k-1}(C) \tag{4}$$

$$R_k(C) = \alpha \times Mpc \times R_{k-1}(\mathrm{PR}) + \beta \times Mcc \times R_{k-1}(C) \tag{5}$$

where *Mpc, Mcp, Mcc* are the propagation matrices, *k* is the times of iterations, α *and* β are the parameters using to adjust the weight of the two network, $\alpha + \beta = 1$.

3.3 PR topic-collaborator relation matrix construction

We construct the corresponding *relation matrix* individually in every project. In this paper, PRs were extracted by GitHub API, and each PR includes title and descriptions. A vector is used to represent each PR which is described by the probability of topics extracted by LDA and labeled with a set of collaborators who reviewed that PR. For applying the LDA on the corpus, we preprocess the text of each PR. We remove stop words from public data set of Google, and restore the rest of the words to a unified tense. In a PR, the probability of a topic indicates the importance of the topic in the PR. The bigger the probability is, the more importance the topic in the PR is. Meanwhile, the probability also can represent the relationship between topics and reviewers. In GitHub, a developer review a PR based on his interest. Generally, the higher the probability is, the stronger the relationship between topic and reviewer is. We calculate multiple topic probabilities (Bian et al. 2014) with PRs that reviewed by the same collaborator, since each collaborator has reviewed a lot of PRs.

Usually, the importance of the topic is different in different PRs, and the importance of the topic is related to the length of the text of PR. The larger the number of text in PR, the greater the importance of the topic should be. Therefore, we define the topic-importance of PR as follows,

$$\mathrm{import}(t_i) = \frac{\sum_d^D N_d P(t_i|d)}{\sum_d^D N_d} \tag{6}$$

where t_i represents the *i*-th topic; *D* indicates the PR set; N_d is the number of words in d-th PR; $P(t_i|d)$ represents topic distribution for PR extracted by LDA algorithm.

However, a collaborator just reviewed some PRs which he or she is interested in. The relationship between a collaborator and topics should be calculated by the part of PRs he or she reviewed. Therefore, we define the relationship between collaborators and topics as follows:

$$R(c_i, t_i) = \frac{\sum_d^D \lambda_{di} N_d P(t_i|d)}{\sum_d^D \lambda_{di} N_d} \tag{7}$$

where c_i is the i-th collaborator; λ_{di} is the control parameter. $\lambda_{di} = 1$ means that this PR is reviewed by the i-th collaborator, otherwise $\lambda_{di} = 0$ denotes no, d represents any document, and D represents the PR set.

Given that the number of PRs reviewed by each collaborator is different, the relationship matrix of topics and collaborators will change due to the activity of the collaborators. More active collaborators have higher topic scores, and less active collaborators have less obvious topical characteristics. To solve this problem, we weight topic score of each collaborator. Therefore, the weight is defined as follows:

$$\mathrm{matrix}(c_i, t_j) = \frac{R(c_i, t_j)}{\sum_k^K R(c_i, t_k)} \tag{8}$$

where *K* is the number of topics in PRs, and t_k represents the *k*-th topic.

3.4 Topic-distribution calculation of new PR

Based on the results calculated in the above steps, the final two steps for recommending the appropriate reviewers for the new PR are to calculate the topic distribution of the new PR and recommend the reviewers for new PR based on the topic and influence score.

Considering the characteristics of the LDA method for extracting text topics, we have two different methods for calculating the topic distribution of the new PR. The first method is to put the text of the new PR into the training set to extract the topic probability, and the other method is to calculate the topic probability of the new PR by using the topic-word distribution extracted by the training set. We assume to have a training data set with 100 PRs and a test data set with 10 PRs. If we use the first method, then we need to run the LDA method and the recommended algorithm for a total of (100 + 1) * 10 times, since each new PR needs to run the LDA method to extract new topics, and the distribution of reviewers and topics also needs to be updated synchronously. If we use the second method, we only need to run the program 100 + 10 times. Obviously, the first method will greatly increase the system time complexity with the number of new PRs. The second method only needs to run the LDA method to extract topic information once, and the accuracy of the topic is only slightly reduced. Based on the comparisons of the above two methods, we take the second way to calculate the topic distribution of new PR.

The second method calculates the subject probability of the new PR using the topic-word distribution generated during the LDA extracting topic process. Therefore, $P(t_i|d)$ is as follows:

$$P(t_i|d) = \frac{\sum_{w \in V} c(w,d)P(w|t_i)}{\sum_k^K \sum_{v \in V} c(v,d)P(v|t_k)} \tag{9}$$

where $c(w,d)$ indicates the number of *w-th* word in *d-th* document, and $P(w|t_i)$ represents the *i-th* topic-word distribution.

3.5 Recommendation reviewer for new PR

Based on the relationship between the topic and the collaborator, the topic probability of the new PR, and the influence score of the collaborator, we can integrate the above steps to recommend the appropriate reviewer for the new PR. If the topic distribution contains multiple maximum values, it means that this PR may be related to multiple collaborators, all of them can be recommended as the reviewers of the PR. As shown in Algorithm 1, NTCRA first obtains the maximum value of the new PR topic. Then, NTCRA matches the collaborator's topic distribution to get the matching score for the new PR and all collaborators. Finally, NTCRA finds the biggest scorer, which is the best core reviewer for the new PR. In case that there are multiple collaborators getting the maximum score, all of them will be recommended for the PR as the candidates of core reviewers.

```
Algorithm I   Core Reviewers Recommendation
Input: Vector of New PR: $v_r$,
       Topic-Collaborator Distribution: $matrix(t_i, c_i)$
       Influence of Collaborator: $influence(c_j)$
1. Topic_PR=list()
2. for r in range(v):
3.     Topic_PR.add(max[topics(v)])
4. Reviewers=list()
5. for r in range(v):
6.   for j in collaborators:
7.     socre(j)=$influence(c_j)$ * $matrix(t_i, c_i)$ *
8.                                 $Topic\_PR(v_r)$
9.     $reviewer(v_r)$.add(socre(j))
10. Reviewers.add(max[$reviewer(v_r)$])
Output: Reviewers
```

Table 1 Detail of datasets

Project	PR	Collaborator	Contributor
fastlane	2779	17	613
mopidy	588	8	93
coala	952	32	209

4 Experiments

4.1 Datasets

We have obtained data from three popular projects from GitHub. The details of the datasets are shown in Table 1. We introduced the PR in the experiment closed on august 1, 2016, and downloaded it via the api provided by GitHub. The api provided by GitHub contains various information data of PR. Usually, we think that the number of developers of a project can roughly reflect the scale of a project. In the following experiments, we have used contributor with similar capabilities to developer as indicator. According to the number of contributors, we selected projects of different sizes as experimental data, as shown in the following Table 1. The scale of a project is defined as follows:

Small: The number of contributors less than 100.

Medium base: The number of contributors more than 100, but less than 500.

Large: The number of contributors more than 500.

4.2 Experiments design

We have proposed four key questions to be solved as follows:

Q1 How about the quality of the influence of collaborator calculation method?

Q2 What is the relationship between topic and collaborator? Is the relationship between topic and collaborator many to many or one to many or others?

Q3 What is the performance of the new PR topic probability calculation method? Can the proposed topic-distribution calculation method be implemented smoothly?

Q4 What about the performance of the proposed method? Is the influence of collaborator performs better than the expertise? And how is the performance changed among the number of topics extracted by the topic model?

For **Q1**, we will compare the influence rank with the expertise rank of collaborators. The expertise of a collaborator is defined as the number of PRs he or she reviewed. Meanwhile, the influence in PR reviewing will be calculated by the proposed method. Through the comparison, we will get how the influence and expertise distribution look like, and get the one which is more suitable for the recommendation.

For **Q2**, we will apply LDA to extract the topics for PRs, and construct the *relation matrix*. Then, we will apply punch card to visualize the *relation matrix*, and recognize the relation pattern between topics and collaborators.

For **Q3**, the divergence of topic distribution by LDA and topic-distribution calculation method of new PRs is measured by Jensen–Shannon divergence (Lin 1991). It is an improved version of the Kullback–Leibler divergence, calculated as

$$D_{js}(p,q) = \frac{1}{2}\left(D_{kl}\left(p,\frac{p+q}{2}\right) + D_{kl}\left(q,\frac{p+q}{2}\right)\right) \tag{10}$$

where *p and q* is the topic distributions of different texts.

For **Q4**, we use the precision and recall to measure the performance of the proposed method. To compare the performance between influence and expertise, we will apply influence and expertise to calculate the precision and recall, respectively. In order to more intuitively and accurately analyze the performance differences between different topics, we chose to experiment with the number of relatively representative topics.

In the topic extraction process, we used JGibbsLDA[1] implemented by Gibbs sampling. We used the default hyper-parameters beta and alpha, and the iteration parameter was set to 1000. Since the LDA topic extraction method is a probability model, it is possible that the returned results are not exactly the same each time. But the distribution of topics is generally consistent, and we value the relationship between different topics and collaborators, rather than the specific content of each topic. For example, we run the LDA method twice and get two topic distributions. It is very likely that the results of the two executions are different. But we can always find the corresponding relationship in the two result sets. And what we need is the relationship between the collaborators and different topics, so although the results are different in different implementations, there is no significant impact on the final recommendation results.

4.3 Evaluation method

We use accuracy and recall to measure the performance of the proposed algorithm in real projects, which have been widely used in the field of recommended systems. According to our definition of core reviewer in section III, the core reviewer of a PR may be more than one. Hence, the precision and recall should be calculated as

$$\text{Precision} = \frac{|\text{act\_coreRev} \cap \text{recom\_coreRev}|}{|\text{recom\_coreRev}|} \tag{11}$$

$$\text{Recall} = \frac{|\text{act\_coreRev} \cap \text{recom\_coreRev}|}{|\text{act\_coreRev}|} \tag{12}$$

where *act_coreRev* is the set of actual core reviewers; The recom_*coreRev* is the set of core reviewers of the PR recommended by the proposed method.

4.4 Model complexity

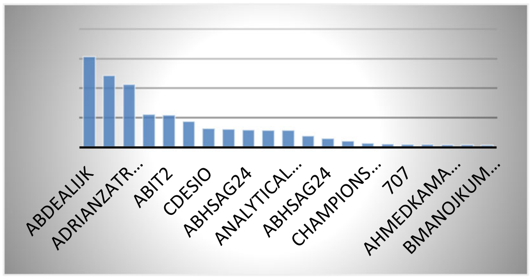
In order to better explain the construction and implementation of the model, we have analyzed the algorithm complexity and time performance of the model. In NTCRA, when we recommend a reviewer for a new PR, we obtain the appropriate reviewer by calculating the product of the reviewer's topic distribution of the subject probability of the new PR and the influence factor of the reviewer. Therefore, the complexity of the algorithm is mainly affected by the number of reviewers, the number of keywords in the new PR, the number of topics, the number of keywords in the training text, and the number of PR in the training data, which can be expressed as $O(wt^q r^{r+q} n)$, where w represents the number of keywords in the training text, t represents the number of topics, q represents the number of PRs in the training data, r represents the number of reviewers, and n represents the number of keywords in the new PR. When NTCRA recommends a reviewer for a new PR, the algorithm obtains the appropriate K reviewers by calculating the match value of each reviewer and the PR. The main time for the program to run is to calculate the matching value of the new PR and each reviewer and its ordering. The average matching time of one reviewer is 36.4 ms (ms).

5 Results

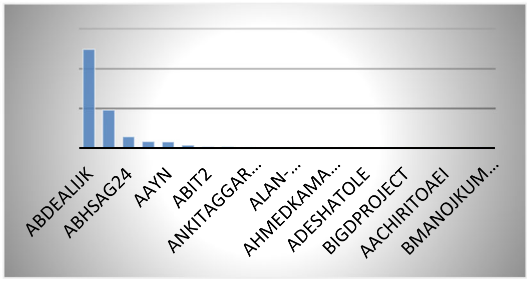
5.1 Collaborator influence

We have constructed the *Collaborator–PR network* and calculated the influence of each collaborator of the project *coala*. When calculating the influence of each collaborator, we have tried multiple group values of α, β. We find that the influence calculation algorithm performs better when setting $\alpha = 0.7$, $\beta = 0.3$. As our expectation, the *Collaborator–PR network* should focus on the review network, supplemented by the common interest network. The common interest network plays a role of balance between active collaborators and less active collaborators. Hence, the result is correct. To verify the performance of the influence calculation algorithm, we conducted a

[1] http://jgibblda.sourceforge.net/.



(a) Influence rank of collaborators



(b) expertise rank of collaborators

Fig. 5 Comparison between influence and expertise of collaborator in project *coala* (**a**) influence rank of collaborators calculated by the algorithm with parameter $\alpha = 0.7$, $\beta = 0.3$; **b** expertise rank calculated by the number of collaborators reviewed, respectively

comparison experiment between influence and expertise. The expertise is calculated by the number of reviewed PRs. The more PRs who reviewed, the higher expertise who will get. Figure 5 shows the results of comparison between influence and expertise. Compared with the expertise, the influence between collaborators performs more balance than the expertise. As shown in Fig. 5b, we can see that the difference of expertise between active collaborators and less active collaborators is extra huge. The expertise of less active collaborator is covered by the active collaborators entirely. As shown in Fig. 5a, there is still a big gap between active collaborators and less active collaborators. But the gap nerve as big as the expertise, it has been reduced extremely. The reduction in gap between active collaborators and less active collaborators will benefit the core-reviewer recommendation very much.

5.2 The structure of relation matrix

For **Q2**, we constructed a *relationship matrix* between collaborators and topics according the proposed method, and showed the results of the relationship matrix on a punch card. As shown in Fig. 6, the punch card shows the *relationship matrix* of collaborators and topics in the *coala* project, and the size of punch reflects the strength of the relationship between the collaborators and the topics. The larger the relationship value between the collaborator and

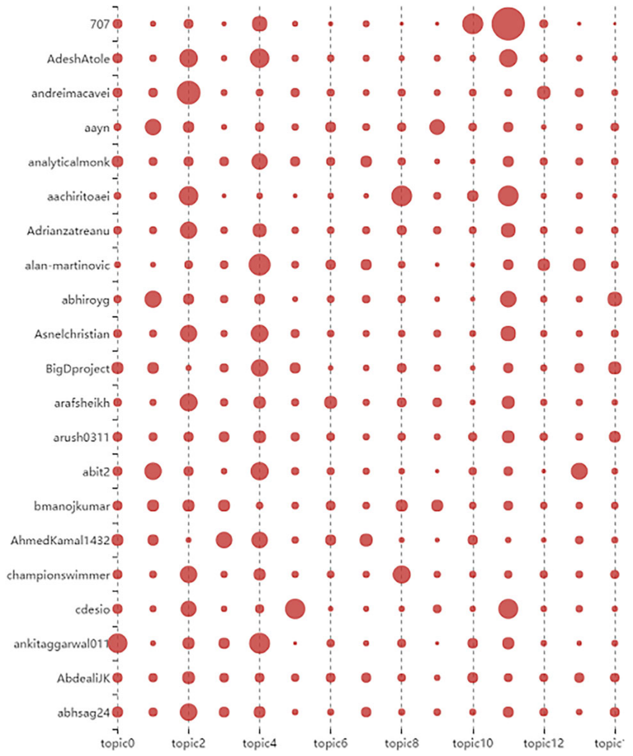


Fig. 6 *Relation matrix* of project *coala* visualized by punch card. The size of punch reflects the strength of the relationship between collaborators and topics. The larger the relationship value between collaborator and topic, the larger the corresponding node on the card

the topic, the larger the corresponding node on the punch card. For the convenience of observation, we just extract 15 topics. The Y-axis denotes the collaborators, the X-axis denotes the topics. From Fig. 6, we can find that the most collaborators always relate to several topics mostly. And the closest topics of different collaborators always are different. According to Fig. 5, we can find punch card of the most active and less active collaborator present very similar. All the size of the punches present nearly. That means that they have the same relationship with all topics. In this case, the influence of collaborators will help recognize the different between them.

5.3 Topic distributions

For **Q3**, we have randomly selected seven PRs, and worked out at the topic distribution by LDA method and the proposed topic-distribution calculation method of new PRs, respectively. To compare the two distributions, we use the stacked histograms to visualize the results. As shown in Fig. 7, the *X*-axis denotes the documents, the *Y*-axis denotes the probability of topics. Each document includes two distributions, the distribution on the left side denotes the distribution calculated by LDA, on the right side one denotes the distribution calculated by the proposed method.

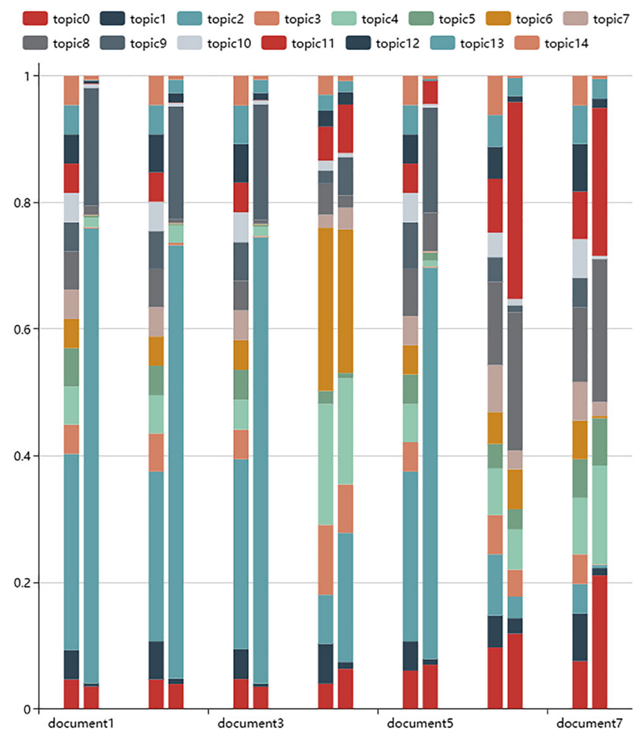


Fig. 7 Topic distributions comparison between LDA and NTCRA. The maximum topics always be consistent of the seven PRs extracted by the two methods

The probability of topics from bottom to top responds topic0–topic14, respectively. From Fig. 7, we can find the maximum entry of the topic distributions between the two methods that tend to be consistent. Since we recommend core reviewer to the new PR based on the topic with the largest value, differences between topics of lower probability do not affect the final recommendation. Moreover, we have selected 1000 comments to separately calculate the topic distribution obtained by LDA and the proposed method, and used Jensen–Shannon divergence to measure the difference between the two results. Jensen–Shannon divergence has a value range of 0 to 1. The smaller the difference between the two distributions, the smaller the value of the Jensen–Shannon divergence. We calculated the Jensen–Shannon divergence in the above 1000 comment data, the average difference is only 0.050. The Jensen–Shannon divergence close to 0 means that the difference between the two distributions is small. Hence, the results show that the proposed method for calculating the topic distribution is effective and feasible.

5.4 Performance

To verify the performance and compare the performance between influence and expertise, we have conducted a comparison experiment between influence and expertise.

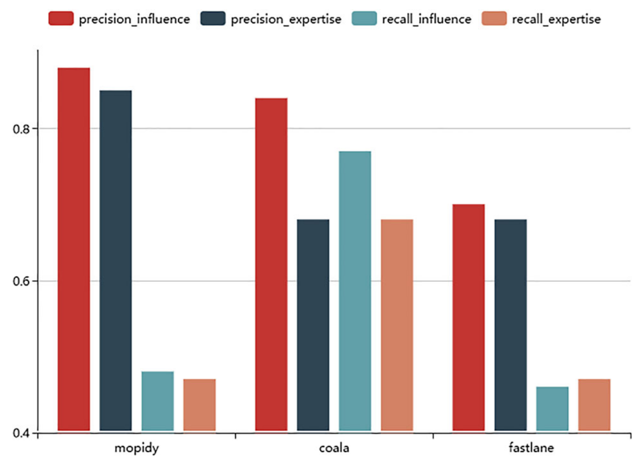


Fig. 8 Performance comparison between influence and expertise of the proposed method. The number of topics extracted is 20. Compared with expertise, the performance has been improved

We apply the proposed method to calculate the performance with influence and expertise on three projects which is shown in Table 1, respectively. Here, the number of topics extracted by LDA is 20. The results are shown in Fig. 8. From the comparison figure, we can find that the influence performs better than the expertise. Even though the recall does not improve in each project, but the precision is improved in each project. Especially in project *coala*, the precision improved significantly. We also can find that the recall in project *fastlane* and *mopidy* is lower than the project *coala*. We check the datasets, and find that there always have more than one collaborators to review the same PR, but the proposed method recommends the collaborator who get the highest score as the core reviewer which means there is just one collaborator matched to the PR. So, the recall is just around 50%.

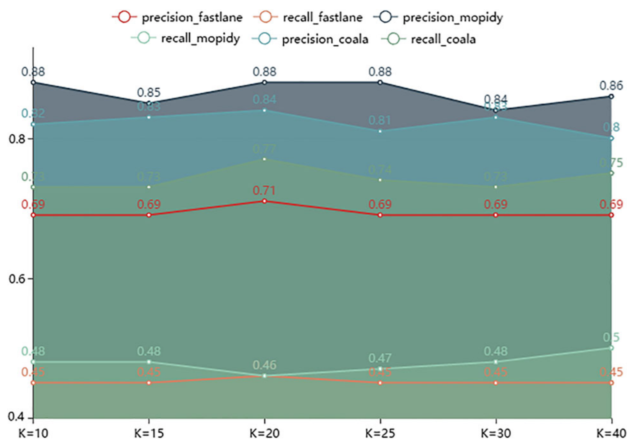


Fig. 9 Performance comparison between different number of topics extracted from PR. Performance calculated with the number of topics = 10, 15, 20, 25, 30, 40, respectively

To explore the influence of number of topics extracted from PRs to the proposed method, we apply the proposed method extracts topics with K(number of topics) = 10, 15, 20, 25, 30, 40, and calculate the precision and recall in each setting, respectively. Figure 9 shows the results. As shown in Fig. 9, in the process of increasing the number of topics from 10 to 40, the recommended accuracy and recall curves of the three projects do not change notably. We also can find the average precision of these three projects is greater than 70%.

6 Discussion and conclusions

6.1 Discussion

Limitation of NTCRA The recall of the proposed method always gets a low score. The proposed method just recommends the collaborator who gets highest score as the core reviewer. Since the mechanism of the proposed method, the number of collaborators who get the highest score is just one in the most situations. However, in some projects, the PR is reviewed by more than one collaborator. Hence, the recall is always less than 50%.

Meanwhile, if there is not enough history data, the algorithm also will get a poor performance.

Frequent matching of active collaborators From the observation, the phenomenon of frequent matching active collaborators as core reviewer still exists, even though the phenomenon reduces a lot comparing with the matching by expertise.

6.2 Conclusions

In this paper, we have proposed an algorithm that recommends a suitable core reviewer for PR, which combines topic model with social network. The NTCRA algorithm uses the text information in PR and comments to build the relationship between collaborators and topics. Meanwhile, we use the review relation of collaborators to construct a collaborator–PR heterogeneous network. Based on the network, we apply a collaborator influence calculation algorithm to calculate the influence of each collaborator to the PRs. Finally, based on the relationship between topics and collaborators, the topic probability of new PR, and the influence score of collaborators, we can integrate the above steps to recommend the appropriate core reviewers for new PR. We have combined the three real open-source projects (*fastlane, coala* and *mopidy*) to study the performance of our algorithm, the influence of reviewer, and the topic distribution. After detailed verification, we have the following conclusions:

(1) The proposed influence calculation algorithm is feasible. The algorithm plays an important role to balance the gap between active collaborators and less active collaborators.
(2) Each collaborator always relates with some closest topics and for different collaborators, the closest topics are different. The most active collaborators may have same relationship to all topics.
(3) The maximum topic calculated by the LDA method is basically the same as the proposed topic-distribution method. The Jensen–Shannon divergence between the two results is only 0.05, and the data indicate that their difference is small.
(4) Overall, the recommended precision of the proposed algorithm is better than 70%, and the change in the number of topics has little effect on the final result.

Even though our algorithm can get an average precision greater than 70%, there still exist a lot of problems. In view of the current low recall rate and fewer core reviewers, we will conduct more detailed analysis and improvement in subsequent studies. We will consider extending the core reviewers to all reviewers in the community, as well as analyzing the reviewer's characteristics and topic preferences to improve the recall rate of the algorithm.

Acknowledgements The works that are described in this paper are supported by NSF 61876190, Ministry of Science and Technology: Key Research and Development Project (2018YFB003800, Hunan provincial 2011 Collaborative Innovation Center for Development and Utilization of Finance and Economics Big Data Property and HNNSF 2018JJ2535).

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

References

Balachandran V (2013) Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In: Proceedings ICSE'13, pp 931–940

Begel A, Bosch J, Storey M-A (2013) Social networking meets software development: perspectives from GitHub, msdn, stack exchange, and top coder. IEEE Softw 30(1):52–66

Bian J, Jiang Z, Chen Q (2014) Research on multi-document summarization based on LDA topic model. In: Proceedings 2014 Sixth international conference on intelligent human-machine systems and cybernetics

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022

Chang X, Ma Z, Lin M, Yang Y, Hauptmann AG (2017) Feature interaction augmented sparse learning for fast kinect motion detection. IEEE Trans Image Process 26(8):3911–3920

Deng Lei, Wang Jiacheng, Zhang Jingpu (2019) Predicting gene ontology function of human MicroRNAs by integrating multiple networks. Front Genet 10:3

Fan X, Chen Z, Cai F, Wu J, Liu S, Liao Z, Liao Z (2018) Local core members aided community structure detection. Mob Netw Appl. https://doi.org/10.1007/s11036-018-0994-2

Gong C, Tao D, Chang X, Yang J (2017) Ensemble teaching for hybrid label propagation. IEEE Trans Cybern 49(2):388–402

Gousios G, Pinzger M, Deursen AV (2014) An exploratory study of the pull-based software development model. In: Proceedings of the 36th international conference on software engineering, ser. ICSE'14. New York, NY, USA: ACM, pp 345–355

Kuang L, Zhu Y, Li S, Yan X, Yan H, Deng S (2018a) A privacy protection model of data publication based on game theory. Secur Commun Netw. https://doi.org/10.1155/2018/3486529

Kuang L, Yu L, Huang L, Wang Y, Ma P, Li C, Zhu Y (2018b) A personalized qos prediction approach for cps service recommendation based on reputation and location-aware collaborative filtering. Sensors 18(5):1556

Kuang L, Yu L, Huang L, Wang Y, Ma P, Li C, Zhu Y (2018c) A personalized qos prediction approach for cps service recommendation based on reputation and location-aware collaborative filtering. Sensors 18(5):1556

Kuang L, He S, Fan Y et al (2019) T-SR: a location privacy protection algorithm based on POI query. IEEE Access 7:59491–59503

Li Y, Wu X, Li L (2015) Community influence analysis based on social network structures. In: IEEE international conference on Smart City/socialcom/sustaincom, pp 247–254

Li Z, Nie F, Chang X, Yang Y (2017) Beyond trace ratio: weighted harmonic mean of trace ratios for multiclass discriminant analysis. IEEE Trans Knowl Data Eng 29(10):2100–2110

Li C, Zheng X, Yang Z, Kuang L (2018) Predicting short-term electricity demand by combining the advantages of ARMA and XGBoost in fog computing environment. Wirel Commun Mobile Comput. https://doi.org/10.1155/2018/5018053

Liao Z, Dayu H, Chen Z, Fan X, Zhang Y, Liu S (2018a) Exploring the characteristics of issue-related behaviors in Github using visualization techniques. IEEE Access 6:24003–24015

Liao Z, Zhao B, Liu S, Jin H, He D, Yang L, Zhang Y, Wu J (2018b) A prediction model of the project life-span in open source software ecosystem. Mob Netw Appl. https://doi.org/10.1007/s11036-018-0993-3

Liao Z, Deng L, Fan X, Zhang Y, Liu H, Qi X, Zhou Y (2018c) Empirical research on the evaluation model and method of sustainability of the open source ecosystem. Symmetry 10:747

Liao Zhifang, Zeng Zhi, Fan Yan Zhang Xiaoping (2019a) A data-driven game theoretic strategy for developers in software crowdsourcing: a case study. Appl Sci 9:721. https://doi.org/10.3390/app9040721

Liao Zhifang, Yi Mengjie, Wang Yan, Liu Shengzong, Liu Hui, Zhang Yan, Zhou Yun (2019b) Healthy or not: a way to predict ecosystem health in GitHub. Symmetry 11:144. https://doi.org/10.3390/sym11020144

Liao Z, Wang N, Liu S, Zhang Y, Liu H, Zhang Q (2019c) Identification-method research for open-source software ecosystems. Symmetry 11(2):182. https://doi.org/10.3390/sym11020182

Lin J (1991) divergence measures based on the Shannon entropy. IEEE Trans Inf Theory 37(1):145–151

Lipcak J, Rossi B (2018) A large-scale study on source code reviewer recommendation. In: 44th Euromicro conference on software engineering and advanced applications (SEAA), pp 378–387

Sayyadi H, Getoor L (2009) FutureRank: ranking scientific articles by predicting their future pagerank. In: Siam international conference on data mining, SDM, Sparks, Nevada, USA, pp 533–544

Thongtanunam P, Kula RG, Cruz AE, Yoshida N, Iida H (2014) Improving code review effectiveness through reviewer recommendations. In: CHASE 2014-proceedings 8th international workshop on cooperative and human aspects of software engineering, pp 119–122

Thung F, Bissyande TF, Lo D et al. (2013) Network structure of social coding in GitHub. In: European conference on software maintenance and reengineering. IEEE, pp 323–326

Xia Z, Sun H, Jiang J, Wang X, Liu X (2017) A hybrid approach to code reviewer recommendation with collaborative filtering. In: International workshop on software mining, pp 24–31. IEEE Computer Society

Yang Z, Huang X, Xiu J et al. (2012) SocialRank: social network influence ranking method. In: IEEE, international conference on cloud computing and intelligent systems. IEEE, pp 591–595

Yang C, Zhang XH, Zeng LB, Fan Q, Wang T, Yu Y, Yin G, Wang HM (2018) RevRec: A two-layer reviewer recommendation algorithm in pull-based development model. J Cent South Univ 25(5):1129–1143

Yu Y, Wang H, Yin G et al. (2014) Reviewer recommender of pull-request in GitHub. In: Proceedings IEEE international conference on software maintenance and evolution (ICSME), 2014, pp 609–612

Zhang Y, Yin G, Yu Y, Wang H (2014) A exploratory study of @-mention in GitHub's pull-requests. In: Proceedings 2014 21st Asia-Pacific software engineering conference

Zhang Y, Yin G, Yu Y, Wang H. (2014) Investigating social media in GitHub's pull-requests: a case study on ruby on rails. In: Proceedings CrowdSoft, Hong Kong, China

Zhang Jingpu, Zhang Zuping, Wang Zixiang, Liu Yuting, Deng Lei (2018) Ontological function annotation of long non-coding RNAs through hierarchical multi-label classification. Bioinformatics 34(10):1750–1757