A Hybrid Approach to Developer Recommendation Based on Multi-relationship

Yifei Da
SKLSDE Lab, School of Computer Science and Engineering
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
Beijing, China
yuantingyuezhi@sina.com

Hailong Sun
SKLSDE Lab, School of Computer Science and Engineering
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
Beijing, China
sunhl@act.buaa.edu.com

Xudong Liu
SKLSDE Lab, School of Computer Science and Engineering
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
Beijing, China
liuxd@act.buaa.edu.com

ABSTRACT

The development of open source community has brought changes to software engineering. Different from traditional software development, developers are free to participate in various projects, QAs and forums under the Internet environment. The quality of participants' contribution largely decides the schedule of projects and the quality of QA answers. However, finding a right participant from numerous website users can be challenging. Therefore, evaluation and recommendation of a person are particularly important. Nonetheless, the developers' multi-relationship is an important behaviour of open source community users, and needs more research work for describing developers' online activities. To this end, we establish a multi-relational network model that describes developers' online behaviours. The capability model and interest model are established for each developer. Furthermore, we utilize a hybrid approach that combines the content and the multi-relationship together to realize the recommendation of respondents for a question. Finally, we have conducted experiments on datasets of blogs, QAs and forum of CSDN. The result proves the effectiveness of our approach.

CCS CONCEPTS

• **Computer systems organization** → **Heterogeneous (hybrid) system**; • **Information Systems** → Database Applications;

KEYWORDS

Software community, recommendation method, developer recommendation, multi-relationship

ACM Reference Format:
Yifei Da, Hailong Sun, and Xudong Liu. 2018. A Hybrid Approach to Developer Recommendation Based on Multi-relationship. In *The Tenth Asia-Pacific Symposium on Internetware (Internetware '18), September 16, 2018, Beijing, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3275219.3275232

1 INTRODUCTION

Software development is highly dependent on developer collaboration. With the development of the Internet, the way developers participate in projects has changed a lot. In the traditional way, the team is fixed. Through the team management, phases of the development process are strictly controlled so that we can gradually meet the needs and improve the products. However, in the Internet environment, more and more projects are completed with GitHub, Stack Overflow, CSDN [1–3]. and other open source communities. At this point, developers voluntarily participate in various projects freely, answer questions on various QA websites and exchange knowledge. Under the Internet environment, on the one hand, team management becomes difficult due to the principle of voluntariness and the quality of participants, for example the ability, interest, and good development habit, becomes more important; On the other hand, developer's quality is very difficult to assess due to the huge number of registered users on the site. Therefore, recommending developers is especially important. Existing work concentrate mainly on the recommendation of the resources [13, 14, 17, 21, 22, 24, 27], which recommends the projects. However, they rarely consider recommending developers. To fill this gap, we conducted following studies: Firstly, we proposed a multi-relationship network based on interaction activities among developers, where the nodes represent developers. Secondly, we established a recommendation algorithm. We proposed to match an unsolved question and developers thus recommend the right developers for questions which need answers. Then we used analysis of the developers' multi-relationship to improve the recommendation. Finally, we evaluated our approach with dataset of blog, QA and forum sections of CSDN, in comparison with state-of-art methods. The results confirmed the advantages of our approach.

2 AN EMPIRICAL STUDY OF CSDN

In real life work, the questions in the QA website are often unanswered, and the projects in out sourcing website are often closed because people could not find the right developer. We did an empirical study to find some reasons for this problem. CSDN is the largest IT community in China and the largest Chinese IT community in the world with more than 30 million registered users. CSDN has

Table 1: Total problems, solved problems and unsolved problems of QA section and OS (out sourcing) section in CSDN

Section	QA section	OS section
Total	137,608	57,890
Solved	36,376	2,339
Solved pro	26.4%	4%
Unsolved	101,232	54,552
Unsolved pro	73.6%	96%

Table 2: Auction number and project budgets of the recent 10 closed projects in CSDN out sourcing section

No.	Auction number	Project budget
1	0	1000-3000
2	10	3000-5000
3	36	>500,000
4	0	<1000
5	0	5000-10,000
6	23	5000-10,000
7	56	100,000-200,000
8	51	50,000-100,000
9	4	1000-3000
10	2	<1000

various sections: blog, forum, QA, out sourcing, etc. We investigated the QA and out sourcing sections to find if people could obtain the need they want through the website.

From Table 1 we can see that, despite great number of people post their problems in CSND to seek solution, most of problems in the website are unsolved. This means the problem posters could no find the person who could and also would like to contribute in these problems. To find the reason, we analyzed the most recently closed 10 projects. The results are in Table 2.

From Table 2, we can see that only for 3 projects (projects 1, 4, 5), the auction number is 0, where there is no people who would like to do the work. This means the project is not attractive enough. But for the other 7 projects, there are always people who are willing to try. Projects 7 and 8 have more than 50 bidders. However, no one is chosen. In the whole outsourcing section, more than 80% closed projects have more than 10 bidders. The similar situation appears in QA section. Some questions have more than 100 answers while none of them is chosen as "best answer".

The conclusions of the empirical study are as follows: Firstly, problems unsolved in the website are numerous and really need to be handled. Secondly, there are many users who are willing to contribute, but the problems and the users are not well matched. In order to deal with this phenomenon, trying to recommend right people for the right problems is extremely important.

3 RELATED WORK

Developers' relationships in the Internet environment are important development behaviors. This has attracted researchers' attention. Zhou et al. [28] established network from projects data, and researched for how developer's communication influenced their contribution. Mockus et al. [20] contributed to evaluate user's capability by researching his experience. These work propose that we can using network to analyze users behavior as well as capability.

For developers blog commenting behavior, the CCTM [5, 6] model defines the behavior of the developer as being "comment-interest" for blogs and commenting under the blog, which combines all developer comments into a single document as an attribute that users are interested in and can comment on a blog. Then they use word embedding to calculate the similarity between blogs and developers [18, 19] and did recommendations based on content. However, this approach is limited in only blog section instead of multi-section content.

For developers answering question behavior on the QA website, the "knowledge exchange network" is a model based on the CSDN forum section [12], which describes accurately developers' knowledge exchange behaviors. The limitation is that it hasn't been used in any recommendation method.

For the developer's participation in the project through an outsourced website, Mark Harman et al. used cheap-talk theory [4, 15, 16] and divided the relationship between the participating developers into two categories: attraction and repulsion. Zizhe Wang, Yang Fu and Luting Ye [11, 23, 26] adopted the user relations in user recommendation in Topcoder which proves to be useful. Zhenglin Xia and Jin Ding [10, 25] applied the same method in Github. These results inspire us of using user relation in user recommendation.

For developers' multi-relationship, for data in the form of triples, the algorithm "TransE" projects the elements of the triples to the same vector space [7–9], using stochastic blockmodel and the projection gets gradually close to the reality. This algorithm is widely used in various situations, and our work will extend the use to improve the recommendation results.

4 CONSTRUCTION OF MULTI-RELATIONSHIP NETWORK

4.1 Definition of Nodes and Edges

In the developers' multi-relationship network, we take the developers as the nodes and the relationships between developers as the edges. The node attributes are user persona of a developer in the website. Defined in 2014, persona is used to collect user information and describe the user character in the Internet. Our persona in this work includes basic attributes as well as a developer interest model and capability model.

For multi-relationship network, we take the developers as the nodes and the relationships between developers as the edges. Firstly we defines network of one single relationship in blog, QA, forum sections, then we combines all the single ones to form a multi-relationship network. The network in each section is as in Figure 1. After this process, we combine six single relationship networks and establish a network of multi-relationships.

Node attributes contain user basic information as well as developer capability model and interest model. The capability model contain a user's published content, because when a user write something, he must have some knowledge about it; developer interest is reflected in the website that browsed by him, because when a user read something, he must be interested by it.

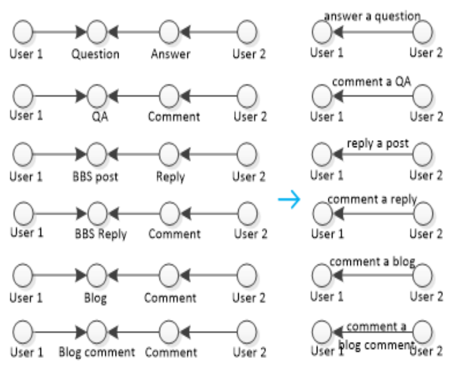

Figure 1: Construction of single relation network.

4.2 Matching of developers and QA tasks

In this process, we mapped questions, user capability, user interest into a same vector space. For a QA task q that needs answers, the literal description of q is mapped to the vector space of the same dimension as the developer interest model and the capability model using the same function. Defining the similarity function between two vectors, then the matching degree between the task q and the developer is defined as the weighted sum of the similarity of developer's capability and the q, combined with the similarity of developer's interest and the q.

5 A HYBRID APPROACH TO DEVELOPER RECOMMENDATION

5.1 Recommendation Based on Multi-Section Content

In the Internet environment development activities, developers can often participate in various activities through multiple sections' content. Researching the diverse content that developers participate in can better portray developer interests and capabilities. This paper uses TF-IDF algorithm for document vectorization. The TF-IDF algorithm is often used for information retrieval, text processing and text similarity calculation. Especially in the case of large amount of text, using TF-IDF to do the calculation of text similarity has proved to be very effective. The cosine similarity algorithm is used as a similarity calculation function. By adjusting the weights, for each problem q to be solved, the similarity between the QA task and each developer can be obtained, and the list is sorted by descending order. Thus a list of developers who have a higher or lower likelihood of content-based recommendations and who are likely to be willing to participate in problem resolution is obtained.

5.2 Recommendation Based on Multi-Relationship

TransE is an accurate and efficient algorithm for dealing with multiple relationships. For data in the form of a triple (h,l,t), h, t are two entities in a multivariate relationship, and l represents the relationship between the two entities. The algorithm projects h, l, t to the same vector space, and then defines the distance formula: Constantly adjust the vector of h, l, t projections to minimize the gap between h+l and t. At this point we can get the projection that best matches the actual situation.

First, set E is defined as a set of all entities. In the training process, the training set is defined as a set of triples, and a set of "trimmed triples" is defined. That is, all other entities in set E are used in place of the actual situation to adjust projections. L-minimum in the loss function:

$$L = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'} [\mathrm{r} + d(h+l,t) - d(h'+l,t')]_{+} \qquad (1)$$

where S is training set S=(sub,rel,obj), and S' is the set of "crashed triples". r can be adjusted to make sure the content within brackets is greater than 0. d is the distance function defined as:

$$d(sub, rel, obj) = \|s + r - o\|_2^2 \qquad (2)$$

Through this process, the distance between h+l and t in the actual situation is minimized, while the distance in all other cases is as large as possible. In the training process, a random gradient descent training method is used, and the parameters are updated in batches in each training in order to achieve the maximum training effect.

After obtaining the best mapping set, the triples consisting of the entity set and the relation set can be predicted. In repeated experiments, we found that the recommended result of the TransE algorithm has the following rules:

1. In some cases, the algorithm has a very high precision rate. The correct result tends to appear in the first few places in the list of results predicted by the algorithm (basically in the top ten).
2. If the correct results do not appear in the top ten, the correct result often appears in the tens or hundreds of digits of the algorithm's predicted result.

This means that the algorithm's prediction of the actual situation does not fully comply with the law of "the higher the rank is, the higher the probability of being ranked", but presents the law of polarization: the prediction could be very accurate or very inaccurate. In principle, the influence of communication behavior between developers presents the law of polarization. In some cases, developers prefer to collaborate with the ones who used to communicate with them; in other cases, the relationship between developers has no impact on their activities in the Internet environment.

5.3 A Hybrid Approach to Results Optimization

While TransE algorithm has proved to have a good effect on the prediction of multi-relational networks, it also has evident disadvantages. We have explained it in 5.2. By contrast, content-based method can largely compensate.. Based on the content-based method, we can obtain the basic result of recommendation. Then we make use of the developer's multiple relationship to optimize the results. It is considered that the top 50 of the content-based recommendation results are of reference value; for the results obtained through the multi-relationship, the top 10 are predicted to be of value. The list that needs to be finally obtained is a user list L of length m and 2<m<10, because in the daily surfing activity, the user reading

ALGORITHM 1: Developer Recommendation Algorithm

```
Input: user list $U$ and $U'$
Output: user list $L$, the lenth of $L$ is m
$i = 0$;
$j = 0$;
$k = 0$;
while $i<50$ and $j<m$ do
    if $U[i]$ is in $U'$ then
        $L[j]= U[i]$;
        $i = i + 1$;
        $j = j + 1$;
    else
        $i = i + 1$;
    end
end
while $k<50$ and $j<m$ do
    if $U[i]$ is in $L$ then
        $L[j]= U[i]$;
        $k = k + 1$;
    end
end
```

recommendation list often ranges from 2 to 10. Defining the result list obtained from 4.1 is:

$$U = \{u_1, ..., u_{50}\} \tag{3}$$

In which we fetch the top 50 developers as the results of reference value. The result list obtained from 4.2 is:

$$U' = \{u'_1, ..., u'_{10}\} \tag{4}$$

In which we fetch the top 10 developers as the results of reference value. Defining the final result list is L, the length of which is m. The process is as in Algorithm 1.

The process of the recommendation algorithm includes two cycles. The first cycle is to find the common recommendation results of U and U'. When the first cycle ends and the final list is less than m, the second cycle is proceeded. The main purpose of the second cycle is to complete the final list till there are m developers in the list.

6 EVALUATION

6.1 Data Collection and Preprocess

The data set used in this article is for CSDN. Founded in 1999, CSDN is the largest IT community in China and the largest Chinese IT community in the world. CSDN is a multi-platform developer community, with various sections such as blogs, forums, questions and answers, outsourcing, downloads, and GitChat. Developers have a distinctive multi-relationship feature.

This article adopts a web crawler method to obtain data and crawls QAs from the first question in 2012 to October 2017, millions of blog data and forum data. The experiment randomly selected 2,000 questions in the CSDN three sections of data. The number of respondents varied from 0 to 79. A total of 1586 developers, 6 relations (comment blog, comment blog comments, answer questions, comment QA, reply BBS posts, comment BBS reply). All relationships are represented in a triple, with a total of 16288 triples. The description of the experimental data set is as Table 3. A random selection of 2,000 questions was used to establish developers' interest and capability models, and build a multi-relationship network. We randomly selected 1000 questions to be the training set and use the rest 1000 to verify the effectiveness of our method.

Table 3: Basic Statistics

Items	Num
users	1586
relations	6
QAs	2000
blogs	1401
BBS posts	2125
triples	16288

Table 4: Experiment results without using multi-relationship, top 3

	CCTM	CCTM extension	Our method
Precision	0.011	0.388	0.392
Recall	0.010	0.375	0.379

6.2 Evaluation Metrics

This experiment mainly contains two parts. The first is to evaluate the effect of multi-relationship of improving the recommendation, the second is to evaluate the final results compared with state-of-art method. Finally, we adjusted the length of the final recommendation list to study the effect of the recommended number on the results.

Precision rate and recall rate are used as evaluation indicators. For a QA task q, the text similarity and user relationship method are used to predict the users that may be used to answer the question. Select the top 3 developers as the final results.

The comparative experiments use the state-of-art method CCTM, and extensions to CCTM. In the extension of CCTM, the contents of developer's attention (ie, the developer interest model in this article) are calculated as part of the content recommendation.

6.3 Experiment Findings and Analysis of Results

We did 2 experiments with our model: In the Table 4, 3 models are tested without considering users' relations. In the Table 5, 3 models are tested with users' relations. E1, E2, E3 represent separately CCTM + relation, CCTM extension + relation, our model + relation.

Seeing from Table 4, the original model of CCTM performed poorly on the CSND dataset. However, after considering the developer's interest to expand, the precision rate immediately increased rapidly. Seeing from Table 5, all the 3 models performed better with the consideration of multi-relationship, especially for the original CCTM model. Moreover, after doing an extension, CCTM performed obviously better, and our method reached the best results.

Table 5: Experiment results using multirelationship

	Top	E1	E2	E3
Precision	1	0.56	0.63	**0.65**
	2	0.28	0.50	0.50
	3	0.18	0.39	**0.40**
	4	0.14	0.32	0.32
	5	0.11	0.27	0.27
	10	0.07	0.15	0.15
Recall	1	0.22	0.22	0.22
	2	0.22	0.32	0.32
	3	0.23	0.38	**0.38**
	4	0.23	0.40	0.41
	5	0.21	0.42	0.43
	10	0.27	0.48	**0.49**
F1 Score	1	0.315	0.326	0.329
	2	0.246	0.390	**0.390**
	3	0.202	0.384	**0.390**
	4	0.174	0.356	0.369
	5	0.151	0.328	0.332
	10	0.111	0.228	0.230

To explain the reason why original CCTM model performs bad in 3 experiment groups. In experiments we found most users doesn't published too much content in the website, so users' latent relationships can barely be reflected in his published content. After considering his interested content, although the proportion of interest model is not high, more users have become able to judge. Therefore, the precision rate increases rapidly. Besides, user relationship judgment can largely compensate for the user who didn't publish too much content in the website, in this case precision and recall rate has increased significantly.

The best precision results appear in the top 1 and the best recall results appear in the top 10 case. In our method, the best precision rate reached 65% and the best recall rate reached 49%. The best F1 score appear in the top 2 and top 3 of our method, which means the method reaches the best performance when recommending 2 or 3 persons.

We also studies the length of the final recommendation list. We tested our method with the recommended list lengths of 1, 2, 3, 4, 5, and 10 respectively. The results are as Figure 2, 3, 4. As can be seen from Figure 2 and Figure 3, the following conclusions can be obtained:

Firstly, when the length of the recommendation list is 1, the model has the best recommendation effect, and the precision rate can reach 65% respectively. Note that when a user satisfies both the "user capability", "user interest" and problem questions at the same time, and the user likes to reply to the questioner, the precision rate of the recommendation is very high. However, this situation can be judged by a small number of people, so when the length of the recommendation list increases, the precision rate decreases rapidly.

Secondly, as the length of the list increases, the recall rate gradually increases. With the expansion of the selection range, the recall rate of the final model can reach 49%. That means, in the best case, half of the right users can be selected by this method.

As in the Figure 4, the F1 scores reach relative high level when the length of recommendation list is 1 to 5. The best results appear when recommending 2 or 3 people. The performance decrease rapidly when the list exceeds 5 people.

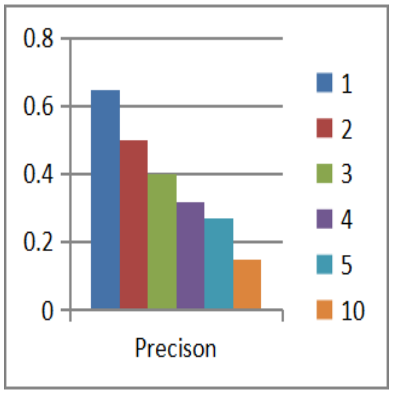

Figure 2: Precision rates with different list lengths.

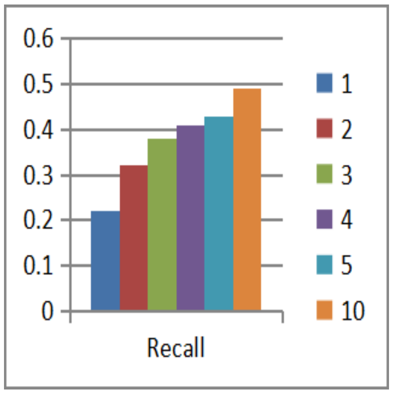

Figure 3: Recall rates with different list lengths.

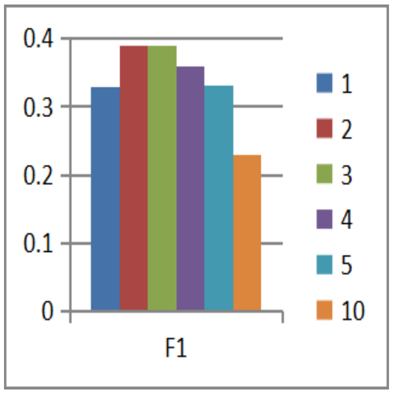

Figure 4: F1 scores with different list lengths .

7 CONCLUSIONS

The vigorous development of the developer community website, after years of accumulation, has become an important channel for programmers to find answers during the process of software development and when they encounter difficult problems. On the one hand, after years of development, the developer community website has accumulated a lot of technical knowledge. On the other hand, publishing blogs, replying to forums, and answering questions reflect the developer's interests, capsabilities, and social situations. By recommending developers, they can greatly promote collaboration in the development of the Internet. We propose a developer recommendation method based on multi-relationship optimization. The single data source recommendation method was extended to multi-section data and optimized using a multi-relationship analysis method. It has been proved that the multi-relationship has improved and promoted the recommendation results. In further research, the algorithm can continually be improved in the following aspects:

Firstly, establishing real-time mechanism to dynamically evaluate user interests and capabilities. In the real world situations, the developer's interests and capabilities have changed over time. Differentiating and analyzing the content released at different times is more in line with the actual situation.

Secondly, finding the weight adjustment method to find the optimal weight. We adjust the weights through repeated experiments. There is still room for improvements.

ACKNOWLEDGMENTS

This work was supported partly by the National Key Research and Development Program of China under Grant No. 2016YFB1000804, and the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2017ZX-14. And the authors would thank the anonymous reviewers for the helpful comments and suggestions to improve this paper

REFERENCES

[1] CSDN website. http://www.csdn.net/
[2] GitHub website. https://github.com/
[3] Stack Overflow website. https://stackoverflow.com/
[4] Nikolay Archak. 2010. Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder. com. In *Proceedings of the 19th international conference on World wide web*. ACM, 21–30.
[5] Trapit Bansal, Mrinal Das, and Chiranjib Bhattacharyya. 2015. Content driven user profiling for comment-worthy recommendations of news and blog articles. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 195–202.
[6] David M Blei and Michael I Jordan. 2003. Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 127–134.
[7] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94, 2 (2014), 233–259.
[8] Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning Structured Embeddings of Knowledge Bases.. In *AAAI*, Vol. 6. 6.
[9] Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618* (2013).
[10] Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-level sentiment analysis of issue comments. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*. ACM, 7–13.
[11] Yang Fu, Hailong Sun, and Luting Ye. 2017. Competition-aware task routing for contest based crowdsourced software development. In *Software Mining (SoftwareMining), 2017 6th International Workshop on*. IEEE, 32–39.
[12] Peng Hongbin and Wang Jun. 2009. Topology of the Knowledge Communication Network in Virtual Communities——Based on CSDN. *Data Analysis and Knowledge Discovery* 25, 4 (2009), 44–49.
[13] Niklas Jakob, Stefan Hagen Weber, Mark Christoph Müller, and Iryna Gurevych. 2009. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*. ACM, 57–64.
[14] Cane WK Leung, Stephen CF Chan, and Fu-lai Chung. 2006. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In *Proceedings of the ECAI 2006 workshop on recommender systems*. 62–66.
[15] Ke Mao, Qing Wang, YUE Jia, and MARK Harman. 2016. Prem: Prestige network enhanced developer-task matching for crowdsourced software development. (2016).
[16] Ke Mao, Ye Yang, Mingshu Li, and Mark Harman. 2013. Pricing crowdsourcing-based software development tasks. In *Proceedings of the 2013 international conference on Software engineering*. IEEE Press, 1205–1208.
[17] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.
[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[20] Audris Mockus and James D Herbsleb. 2002. Expertise browser: a quantitative approach to identifying expertise. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*. IEEE, 503–512.
[21] Nikolaos Pappas and Andrei Popescu-Belis. 2013. Sentiment analysis of user comments for one-class collaborative filtering over ted talks. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 773–776.
[22] Štefan Pero and Tomáš Horváth. 2013. Opinion-driven matrix factorization for rating prediction. In *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 1–13.
[23] Zizhe Wang, Hailong Sun, Yang Fu, and Luting Ye. 2017. Recommending crowdsourced software developers in consideration of skill improvement. In *Automated Software Engineering (ASE), 2017 32nd IEEE/ACM International Conference on*. IEEE, 717–722.
[24] Yao Wu and Martin Ester. 2015. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 199–208.
[25] Zhenglin Xia, Hailong Sun, Jing Jiang, Xu Wang, and Xudong Liu. 2017. A hybrid approach to code reviewer recommendation with collaborative filtering. In *Software Mining (SoftwareMining), 2017 6th International Workshop on*. IEEE, 24–31.
[26] Luting Ye, Hailong Sun, Xu Wang, and Jiaruijue Wang. 2018. Personalized teammate recommendation for crowdsourced software developers. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 808–813.
[27] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 83–92.
[28] Minghui Zhou and Audris Mockus. 2011. Does the initial environment impact the future of developers?. In *Proceedings of the 33rd International Conference on Software Engineering*. ACM, 271–280.