Suggesting Questions that Match Each User's Expertise in Community Question and Answering Services

Katsunori Fukui
Graduate School of Systems Engineering
Wakayama University
Wakayama, JAPAN
fukui.katsunori@g.wakayama-u.jp

Tomoki Miyazaki
Graduate School of Systems Engineering
Wakayama University
Wakayama, JAPAN
tomoki.miyazaki@g.wakayama-u.jp

Masao Ohira
Graduate School of Systems Engineering
Wakayama University
Wakayama, JAPAN
masao@wakayama-u.ac.jp

***Abstract*—Stack Overflow has been already recognized as an indispensable CQA (Community Question and Answering) service for developers. However developers as questioners sometimes cannot get any answers or they only can get incomplete answers to resolve their questions, since over several thousands of questions and answers are posted on a daily basis and expert developers are currently facing with a difficulty in finding questions which can be better or best answered by them. In order address the issue, our study aims at developing a bot that helps expert developers find questions matching each developer's expertise. The existing study [8] proposed a promising approach to the expert recommendation for CQA services, based on PMF (Probabilistic Matrix Factorization) algorithm. In order to improve the accuracy of the expert recommendation, in this paper we present our approach which combines the PMF (Probabilistic Matrix Factorization) approach and a term expansion techniques using word embeddings. As a result of an experiment comparing our approach with the existing approach only based on PMF, we show that our approach based on PMF and the term expansion with Word2vec and fastText slightly outperforms the existing approach only based on PMF.**

***Index Terms*—expert recommendation, community question and answering (QCA) services, Stack Overflow, PMF (Probabilistic Matrix Factorization), term expansion, word embeddings, Word2vec, fastText, personal bot for software engineering**

I. Introduction

Since modern software products often become large and complex to satisfy various, changing user needs during long-term use, developers need to acquire technical information and knowledge from outside developers [1]. For that reason, communication media for discussions among developers have changed from mailing lists or forums in a project to community question and answering services (CQA) such as Stack Overflow in recent years.

In Stack Overflow, over nine million users discuss a wide variety of topics relative to computing technologies such as programming languages and development tools. Stack Overflow has been already recognized as an indispensable CQA service for developers, but an important issue has been arising from its popularity. In Stack Overflow, over several thousands of questions and answers are posted on a daily basis. However, about half of the questions are not resolved and about 30% of the questions are not even answered [2]. Furthermore, unresolved questions tend to be increasing these days [3].

Users in Stack Overflow can find questions from "Top Questions" which show the latest, featured, or most viewed questions on the top page of Stack Overflow. Otherwise, they can search questions with queries or keywords tagged by questioners. Due to the current search mechanism provided by Stack Overflow and a large number of posted questions as mentioned above, users are currently facing with a difficulty in finding questions which can be better or best answered by them. As a result, developers as questioners sometimes cannot get any answers or they only can get incomplete answers to resolve their questions.

Several existing studies [4]–[8] proposed methods to recommend experts (i.e., appropriate questionees for each question) to address the issue above. If the proposed methods are applied to CQA services, users would not need to search questions anymore and each user can get a notice from the CQA services individually when a question is posted and it matches expertise of each user. Our study is also in line with the existing approach, but we are trying to improve the recommendation accuracy using "term expansion" described later and implementing a recommendation engine as a personal bot in order to help users in CQA services find questions more effectively.

In what follows, existing studies are introduced and the limitation of them is discussed in Section II. Our approach using the term expansion is presented in Section III. A preliminary experiment and its result are described in Section IV. Based on the result of the experiment, the future direction of our study is discussed in Section V. Finally the paper is concluded and our future work is remarked in Section VI

II. Expert Recommendation in CQA

Although there are various text mining approaches to supporting users in CQA services such as best answer prediction [9], [10] which helps questioners select the best answer when many answers are posted (i.e., less knowledgeable questioners cannot judge which is the best answer.), in this section we focus on some existing studies on expert recommendation which aims at helping expert developers (questionees) find questions that match their own expertise and at the same time aims at resulting in helping regular developers (questioners) get useful answers in CQA services.

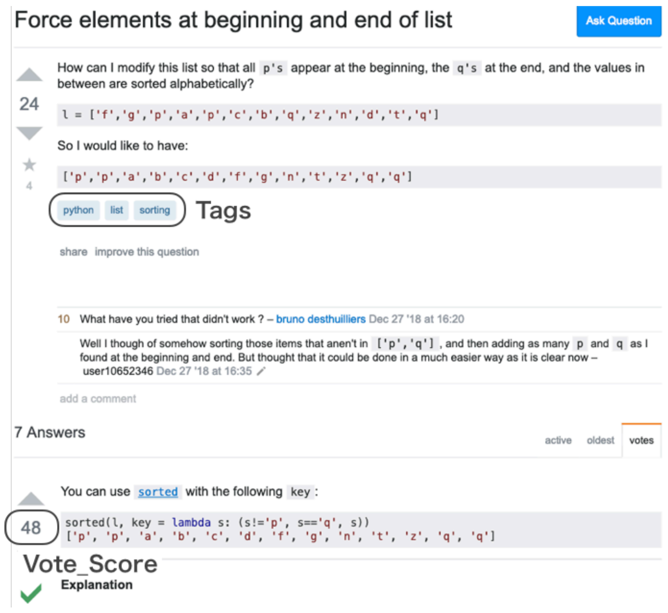
A. *Topic Modeling*

One of promising approaches to the expert recommendation in CQA is to use topic modeling which assists to infer "*who is knowledgeable about which topic?*" from relationships between topics and and past answers relative to the topics. Yang et al. [4] proposed Topic Expertise Model (TEM) using topic modeling and also proposed a method called CQArank which combines TEM and link structures among users to recommend experts. Wang et al. [5] proposed Question-LDA based on Twitter-LDA which can extract topics even if a text consists of a small number of terms (i.e., short text such as a tweet). They also proposed an expert recommendation method called NEWHITS which improves the HITS algorithm used to consider link structures among users. However, the topic modeling-based approaches requires all of the question and answer data such as texts and keyword tags to extract topics. In other words, an enormous amount of computing resource is required to apply the proposed methods to CQA services, since CQA services such as Stack Overflow store a vast number of questions and answers. It would be very difficult to constantly or periodically apply the proposed methods (i.e., create a topic model) to CQA services.

In addition to the topic modeling approaches above, some improved techniques have been proposed. Tondulkar et. al [6] focus not only on text information of questions and answers but also on tag information and historical information of users' activities in CQA services and used LamdbaMART [11] as a rank learning algorithm to rank users for a posted question. Wang et. al [7] proposed an expert recommendation approach based on CNN (Convolutional Neural Networks). It utilizes text information of questions and answers and information of user profiles. As same as the topic modeling approaches, these approaches uses a vast number of text (questions and answers), tags and users' activities and profiles. Since our study aims at improving the prediction accuracy of tag-based approaches which direct effort toward reducing computational costs, the existing studies introduced in this subsection are different approaches from our study.

B. *Tag-based Recommendation*

To reduce the computational cost for topic modeling, Yang et al. [8] proposed an expert recommendation method which dose not use contents of questions and answers but only uses keywords tagged with questions. In the study, tagged keywords are assumed to well-summarize contents of a question and to act as "topics" as same as topic modeling. The proposed method firstly extracts tagged keywords in a question and voting scores in an answer as illustrated in Figure 1. In Stack Overflow, users can vote answers that are useful for the question. In the study, higher voting scores are assumed



Force elements at beginning and end of list

How can I modify this list so that all `p's` appear at the beginning, the `q's` at the end, and the values in between are sorted alphabetically?

```
l = ['f','g','p','a','p','c','b','q','z','n','d','t','q']
```

So I would like to have:

```
['p','p','a','b','c','d','f','g','n','t','z','q','q']
```

10 What have you tried that didn't work ? – bruno desthuilliers Dec 27 '18 at 16:20

Well I though of somehow sorting those items that anen't in `['p','q']`, and then adding as many `p` and `q` as I found at the beginning and end. But thought that it could be done in a much easier way as it is clear now – user10652346 Dec 27 '18 at 16:35

add a comment

You can use `sorted` with the following `key`:

```
sorted(l, key = lambda s: (s!='p', s=='q', s))
['p', 'p', 'a', 'b', 'c', 'd', 'f', 'g', 'n', 't', 'z', 'q', 'q']
```

Fig. 1. Example question in Stack Overflow. The question is tagged with the three keywords: python, list and sorting. The question has severn answers and one of the answers is voted by forty eight users as "This answer is useful."

to be a more useful answer and the voting score is used as an indicator to evaluate the quality of the answer.

Second, the proposed method creates a user-tag expertise matrix where the row consists of users U_i and the column consists of tagged keywords T_j. A value in the matrix are calculated by averaging voting scores of U_i's answers to questions with T_j. Here, since it is not possible that all the users answer all the questions, the original matrix becomes sparse. However, it does not necessarily mean a user cannot answer questions with particular keywords which have not been used for questions s/he has answered. In the study, PMF (Probabilistic Matrix Factorization) [12] is utilized to predict and complement zero values in the sparse matrix. In other words, PMF is used to infer the voting score when a user answers a question which have not been answered by the user.

Finally, using the user-tag expertise matrix processed by PMF, the proposed method recommends experts suitable for a newly posted question. When a new question is posed, tagged keywords are extracted, voting scores for U_i are calculated, and then users having higher scores are recommended for the question. In [8], the proposed method shows better performance in the recommendation accuracy and in the running time than the existing topic modeling-based method (TEM).

C. *Limitation of Tag-based Recommendation*

Although [8] outperforms the existing method, we believe it can still be improved. The expert recommendation method in [8] does not consider spelling variants of tagged keywords and similar keywords but deals with them individually. For instance, suppose that "java" is tagged to one question and "java-9" is tagged to another question when two questions are posted and that a developer answers the question with "java-

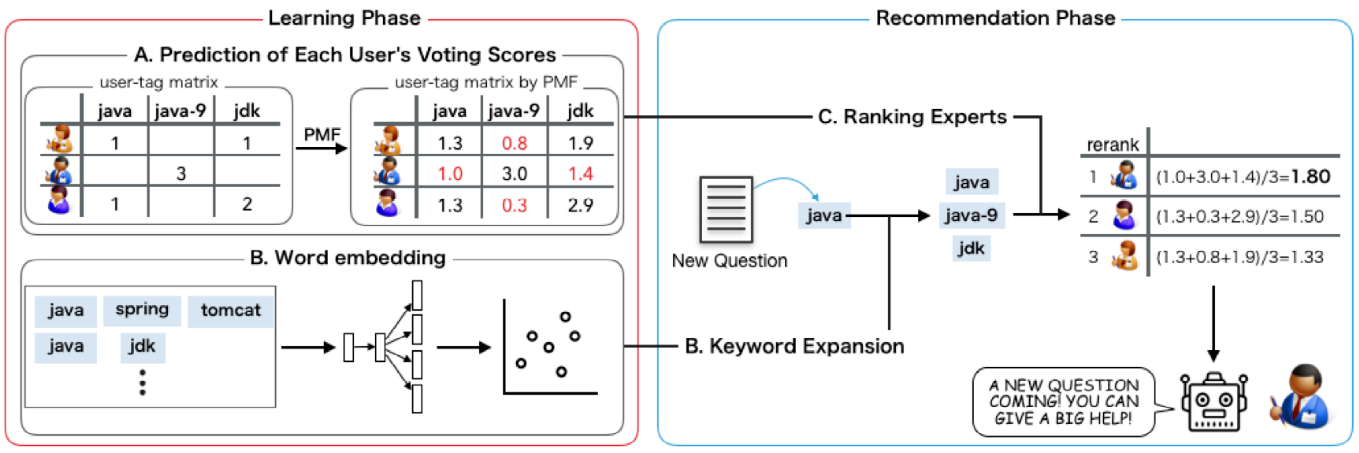


Fig. 2. Overview of our approach. It consist of three parts: (A) prediction of each user's voting scores using PMF [12], (B) keyword expansion using word embeddings (Word2vec or fastText), and (C) ranking for the expert recommendation.

9" and had not answered questions with "java." In this case, the developer's voting score for questions with "java become zero even if the developer has a great deal of expertise on "java." In our study, we try to improve [8] by expanding tagged keywords based on word embeddings in order to mitigate the issue of spelling variants of tagged keywords. In the near future, we will implement the improved method as an engine for a personal bot to help an expert developer find questions that match her/his expertise. In the next section, we present our approach in detail.

III. Approach

Figure 2 shows an overview of our approach which consists of three parts: (A) prediction of each user's voting scores using PMF [12], (B) keyword expansion using word embeddings, and (C) ranking for the expert recommendation. In what follows, we describe each part.

A. Prediction of Each User's Voting Scores using PMF

As same as [8], a user-tag expertise matrix is firstly created based on each user's voting scores and tagged keywords used in questions. An entry s_{ij} in the matrix represents an average score for user u's answers to questions with a tagged keyword t. Next, PMF is applied to the matrix to predict and complement voting scores in the matrix as illustrated in the top left of Figure 2.

B. Keyword Expansion using Word Embeddings

As we mentioned earlier, the existing approach [8] does not consider spelling variants of tagged keywords and similar keywords but deals with them individually. In order to address the issue that makes a user-tag expertise matrix sparse (i.e., the limitation of the tag-based recommendation approach), our approach uses term expansion based on word embeddings. Word embedding is a technique to obtain a word vector space model based on neural networks. In this study, we use the skip-gram models in Word2vec [13] and fastText [14] respectively. In our approach, keywords tagged for a past question are extracted and are regarded as a set of tagged keywords. Furthermore, a set of tagged keywords is treated as a sentence and all the sets of tagged keywords extracted from past questions in the target dataset are modeled based on word embeddings.

The same keywords would be positioned around similar keywords in the word vector space models. For instance, suppose that a new question is tagged with "java-9" and one expert had answered questions tagged with "java" but had not answered to "java-9." In our approach, the term "java-9" can be expanded to "java-9" and "java" since "java-9" and "java' should be nearly positioned in the representation of word embeddings.

The number of expanded terms (keywords) can be arbitrarily changed to optimize the prediction accuracy. For instance, if a question has a tagged keyword "java-9," the number of expanded terms can be changed such as [java-9, java (Expand 1)], [java-9, java, java-module (Expand 2)], [java-9, java, java-module, jigsaw (Expand 3)], and so on. The term expansion is applied to each keyword which is originally tagged to a question. PMF also works as with the term expansion since it complements voting scores for questions which have not been answered by experts. Our approach uses the term expansion to strengthen the effect of PMF using tagged keywords extracted from a newly posted question.

C. Ranking Experts for a Newly Posted Question

For a newly posted question q, the recommendation score $ReScore(u,q)$ is calculated as

$$ReScore(u,q) = \frac{1}{N_t}\sum_{t=0}^{N_t} R(u,t) \tag{1}$$

where $R(u,t)$ represents the recommendation score of user u for tagged keyword t which is calculated by PMF. N_t is the number of keywords tagged for the new question q.

$ReScore(u,q)$ means the average score of PMF when user u answers a question q with some tagged keywords.

Based on the calculated recommendation scores for all the users, a list of ranked experts can be created as illustrated in the right of Figure 2. Although the existing study only uses the list to find experts suitable for a new question, a bot based on our approach uses the list to let promising experts know when the question suitable to the experts is coming.

IV. EXPERIMENT

To evaluate our approach, we conduct a preliminary experiment which compares our approach with the existing approach [8].

A. Dataset

For the experiment, questions and answers posted from 2008 to 2017 are collected from Stack Overflow in a XML format as a dataset. Extracting tagged keywords used for questions from July 31, 2008 to December 31, 2014, two word embedding models are created using Word2vec and fastText respectively. For PMF's learning data, tagged keywords from January 1, 2015 to December 31, 2015 are extracted. Voting scores are also extracted only from users who answered over 50 times during the same period. For evaluating our approach and the existing approach, we select questions which are answered by more than six users and their answers from January 1, 2016 to December 31, 2017.

B. Experiment Setting

In the experiment, the existing approach (PMF) and our approach (PMF + term expansion) are compared using the recommendation accuracy. The recommendation accuracy is calculated using nDCG (normalized discounted cumulative gain) [15] which is regularly used as a performance indicator in ranking recommendation studies and indicates how recommended ranked data is adequate. nDCG is expressed from 0 to 1 and the higher nDCG score means the better performance. As our approach has the term expansion feature, we evaluate our approach changing the number of expanded keywords from one to four. For each approach, nDCG is calculated ten times and the average score of them are presented as a result because PMF stochastically predicts and complements values of entities (voting scores) in the user-tag expertise matrix.

The experiment is run on Macbook Pro (CPU: 2.5GHz Intel Core i7, Memory: 16GB). It takes about two seconds for each iteration and about a thousand seconds for 500 iterations in total for creating a PMF model which is used in the both approaches. Creating a vector model for Word2vec and fastText respectively spends about one hour and four hours for 100 iterations.

C. Result

1) nDCG score: Table I shows the result of the experiment. The result of the existing approach is used as baseline (0.824). From the table, we can confirm our approach using Word2vec outperforms the existing approach in case where additional one (Expand 1) and two (Expand 2) keywords are added to the original keywords tagged to a question. However, the term expansion does not improve the nDCG score in case of using additional three (Expand 3) and four (Expand 4) keywords. On the other hand, our approach using fastText outperforms the existing approach in all the conditions. In particular, one additional keyword by fastText (Expand 1) shows the best nDCG score (**0.833**).

TABLE I
RESULT (NDCG SCORE)

Method	nDCG
Baseline	0.824
Word2vec (Expand 1)	**0.825**
Word2vec (Expand 2)	**0.827**
Word2vec (Expand 3)	0.822
Word2vec (Expand 4)	0.823
fastText (Expand 1)	**0.833**
fastText (Expand 2)	**0.825**
fastText (Expand 3)	**0.827**
fastText (Expand 4)	**0.831**

TABLE II
RANK RESULT (QUESTION ID: 41960019)

User ID	Ideal	Baseline	fastText
5047996	1	2	2
2861476	2	4	1
778560	2	5	3
388389	3	6	4
1417694	3	1	5
2128947	3	3	6
nDCG	1.000	0.766	0.899

Term Expansion: batch-file→command-prompt, cmd→command-prompt

TABLE III
RANK RESULT (QUESTION ID: 41613710)

User ID	Ideal	Baseline	fastText
3001626	1	8	3
4341440	2	6	7
3304471	3	1	1
3987294	4	2	2
5635580	4	3	6
3732271	5	5	4
3962914	5	7	8
nDCG	1.000	0.821	0.871

Term Expansion: r→wolfram-mathematica

2) Examples of Ranking Results: As we mentioned earlier, a nDCG score in Table I is the average of nDCG scores calculated ten times for each question. Although Table I shows fastText (Expand 1) indicates the best nDCG score in this experiment, the nDCG score varies depending on each question. In other words, nDCG in one condition (e.g., fastText (Expand 1)) is not always consistent for all the questions but may shows better and worse scores. To better understand the effect of the term expansion, we select and analyze examples of two better rankings and two worse rankings in the condition of fastText (Expand 1).

Table II and Table III shows examples of rankings when nDCG scores outperform the average score (**0.833**). In the

TABLE IV
RANK RESULT (QUESTION ID: 42239179)

User ID	Ideal	Baseline	fastText
4653379	1	2	4
5291015	2	1	6
2173773	3	7	7
589924	4	3	5
6862601	4	6	2
298607	5	4	3
nDCG	1.000	0.960	0.705

Term Expansion: awk→nawk, bash→bash4, grep→find-grep, linux→linuxbrew, perl→perlguts

TABLE V
RANK RESULT (QUESTION ID: 46043445)

User ID	Ideal	Baseline	fastText
3063910	1	1	4
4895725	2	2	3
3732271	3	3	3
3604745	3	3	3
496488	4	3	2
nDCG	1.000	0.836	0.695

Term Expansion: appned→prepend, dataframe→data.table, matrix→matrix-math, r→wolfram-mathematica, rbind→cbinda

Tables, "Ideal" means a result of the actual ranking based on actual voting scores. "Baseline" and "fastText" means a result of the predicted ranking based on the existing approach (only PMF) and our approach (PMF + term expansion) respectively. Note that our approach in this comparison uses fastText (Expand 1).

As shown in Table II, fastText (Expand 1) is much better (0.899) than Baseline (0.766) for Question ID: 41960019. Question ID: 41960019 is titled "*count an exact character in one line - cmd*" with two tags ("batch-file" and "cmd") originally and is answered by six experts. 41960019 is about how to write a batch file to count the number of occurrences of a specific character in each line of a text file. fastText (Expand 1) added a tag "command-prompt" to the original tags ("batch-file" and "cmd"). Compared to Ideal (ranking based on actual voting scores), fastText (Expand 1) only have a difference of ranks between top 1 and top 2, while Baseline is very different from Ideal. This is a good example that the term expansion improved the accuracy of PMF.

As shown in Table III, fastText (Expand 1) is slightly better (0.871) than Baseline (0.821) for Question ID: 41613710. Question ID: 41613710 is titled "*Extend an irregular sequence and add zeros to missing values*" with one tag ("r") originally and is answered by seven experts[1]. 41613710 is about how to expand an irregular sequence with a regular sequence in a data frame using R. fastText (Expand 1) added a tag "wolfram-mathematica" to the original tag ("r"). Compared to Ideal, Baseline ranked top 8 in Ideal as top 1, while fastText (Expand 1) ranked top 8 in Ideal as top 8. However, the ranking of fastText (Expand 1) also still seems imperfect since top 7 is ranked as top 2 and top 1 and top2 are not improved from Baseline. From this example, we cannot clearly confirm the effect of the term expansion.

Table IV and Table V show examples of rankings when nDCG scores underperform the average score (**0.833**). As shown in Table IV and Table V, the nDCG scores of fastText (Expand 1) are much worse (0.705 and 0.695) than Baseline (0.960 and 0.836) respectively for Question ID: 42239179 and 46043445.

Question ID: 42239179 is titled "*Fastest way to find lines of a file from another larger file in Bash*" with five tags ("awk", "bash", "grep", "linux" and "perl") originally and is answered by six experts[2]. Question ID: 46043445 is titled "*How to append group row into dataframe*" with five tags ("appned", "dataframe", "matrix", "r" and "rbind") originally and is answered by five experts[3]. Both the results of fastText (Expand 1) are very different from Ideal and Baseline. This might happen because both the questions respectively have five tags and our approach uses five original tags and five expanded tags for each question (i.e., over-expansion). Although fastText (Expand 1) only expands each original tag to two tags (original + expanded tags), the term expansion technique have a bad effect on the prediction accuracy when a question has many tags such as Question ID: 42239179 and 46043445. This might be also related to the reason why Word2vec (Expand 3) and Word2vec (Expand 4) were worse than the baseline and why fastText (Expand 2), fastText (Expand 3), and fastText (Expand 4) were worse than fastText (Expand 1) in Table I. We need to further investigate the effect of the term expansion in the future.

V. DISCUSSIONS

From the experiment, we found that our approach using the term expansion slightly outperforms the existing approach and the term expansion based on fastText works better than Word2vec. The reason why the term expansion based on fastText is better than Word2vec is that fastText uses information of subwords which are a piece of a word. In general, it is well known that Word2vec cannot precisely obtain synonyms for one word when the word is not frequently appeared in text documents. On the other hand, since fastText learn text data using subwords, the term expansion based on fastText would contribute to obtain similar keywords even from rarely appeared keywords. fastText can also obtain similar keywords which are only included in test data but not included in learning data because of the same reason above (e.g., since keyword "sqlite" not appeared in learning data is divided to sql+qli+lit+ite, similar keywords such as "mysql" and "sql" can be obtained from test data.) , but Word2vec cannot do so.

Although in this paper the term expansion based on word embeddings is assumed to be applied after receiving a new question, we are planing an alternative approach which does

[1]This question was actually answered by nine users. Because of the filtering described in subsection IV-A, the seven users with over 50 answers in the past were selected as experts.

[2]This question was actually answered by sixteen users.

[3]This question was actually answered by nine users.

not use the term expansion but creates a user-tag expertise matrix based on PMF after unifying similar tagged keywords using word embeddings. Unifying similar tagged keywords in advance might contribute to the improvement in the prediction accuracy and the reduction of the computational cost for PMF since it transforms rarely used keywords (i.e., noisy data for the prediction) into frequently used similar keywords. We also plan to further improve our approach in order to reduce the computational cost for word embeddings using Word2vec and fastText. Currently all the tagged keywords are learned by Word2vec and fastText, but the computational cost must be reduced by only learning commonly appeared keywords using association rule mining. It also might contribute to the improvement in the prediction accuracy because it means rarely used keywords are not learned by Word2vec and fastText.

VI. Conclusion and Future Work

In this paper we present our approach to the expert recommendation for CQA services which are actively used by an enormous number of developers engaging in the modern software development. Our approach based on PMF and the term expansion with Word2vec and fastText slightly outperforms the existing approach only based on PMF.

We are currently working to incorporate our approach into an engine for a personal bot which helps expert developer find questions that match their own expertises. At the same time, as we discussed above, we are studying to improve the prediction accuracy and to reduce the computational costs for PMF and word embeddings.

Acknowledgment

This research is conducted as part of Grant-in-Aid for Japan Society for the Promotion of Scientific Research (17H00731, 18K11243).

References

[1] B. Vasilescu, A. Serebrenik, P. Devanbu, and V. Filkov, "How social Q&A sites are changing knowledge sharing in open source software communities," in *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*, 2014, pp. 342–354.

[6] R. Tondulkar, M. Dubey, and M. S. Desarkar, "Get me the best: predicting best answerers in community question answering sites," in *Proceedings of the 2018 ACM International Conference on Recommender Systems (RecSys '16)*, ser. RecSys '18. ACM, 2018, pp. 251–259.

[2] L. Nie, X. Wei, D. Zhang, X. Wang, Z. Gao, and Y. Yang, "Data-driven answer selection in community QA systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1186–1198, 2017.

[3] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, "Answering questions about unanswered questions of Stack Overflow," in *Proceedings of IEEE International Working Conference on Mining Software Repositories (MSR '13)*, 2013, pp. 97–100.

[4] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen, "CQArank: Jointly Model Topics and Expertise in Community Question Answering," in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)*, 2013, pp. 99–108.

[5] L. Wang, B. Wu, J. Yang, and S. Peng, "Personalized recommendation for new questions in community question answering," in *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '16)*, ser. ASONAM '16. IEEE Press, 2016, pp. 901–908.

[7] J. Wang, J. Sun, H. Lin, H. Dong, and S. Zhang, "Convolutional neural networks for expert recommendation in community question answering," *Science China Information Sciences*, vol. 60, 2017.

[8] S. M. Yang Baoguo, "Tag-based expert recommendation in community question answering," in *Proceedings of 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '14)*, 2014, pp. 960–963.

[9] F. Calefato, F. Lanubile, and N. Novielli, "Moving to Stack Overflow," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '16)*, 2016, pp. 1–10.

[10] ——, "An empirical assessment of best-answer prediction models in technical Q&A sites," *Empirical Software Engineering*, 2018.

[11] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," Microsoft Research, Technical Report MSR-TR-2010-82, 2010.

[12] S. Ruslan and M. Andriy, "Probabilistic Matrix Factorization Ruslan," in *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS '07)*, 2007, pp. 1257–1264.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *CoRR*, vol. abs/1607.04606, 2016.

[15] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.