Predicting Best Answerers for New Questions: An Approach Leveraging Topic Modeling and Collaborative Voting

Yuan Tian, Pavneet Singh Kochhar, Ee-Peng Lim, Feida Zhu, and David Lo

Singapore Management University, Singapore
{yuan.tian.2012,kochharps.2012,eplim,fdzhu,davidlo}@smu.edu.sg

Abstract. Community Question Answering (CQA) sites are becoming increasingly important source of information where users can share knowledge on various topics. Although these platforms bring new opportunities for users to seek help or provide solutions, they also pose many challenges with the ever growing size of the community. The sheer number of questions posted everyday motivates the problem of routing questions to the appropriate users who can answer them. In this paper, we propose an approach to predict the best answerer for a new question on CQA site. Our approach considers both user interest and user expertise relevant to the topics of the given question. A user's interests on various topics are learned by applying topic modeling to previous questions answered by the user, while the user's expertise is learned by leveraging collaborative voting mechanism of CQA sites. We have applied our model on a dataset extracted from StackOverflow, one of the biggest CQA sites. The results show that our approach outperforms the TF-IDF based approach.

Keywords: CQA, expert recommendation, topic modeling, collaborative voting.

1 Introduction

Community Question Answering (CQA) sites archive millions of questions and answers posted by users. General CQA sites like Yahoo! Answers[1] and Quora[2], and domain-specific CQA sites like StackOverflow[3] and Mathematics[4] have attracted millions of users. CQA sites provide platforms for users to collaborate in the form of asking questions or giving answers. The main purpose of such communities is to provide high quality answers[5] and to offer a wide variety of

[1] `http://answers.yahoo.com/`
[2] `https://www.quora.com/`
[3] `http://stackoverflow.com`
[4] `math.stackexchange.com`
[5] High quality answers are the answers that satisfy question asker [1] and other web users who face similar problem in the future [2].

solutions or explanations. To maintain high quality questions and answers, most of the CQA sites use collaborative voting mechanism, which allows users inside the community to vote up or down the questions and the answers that they like or dislike. These question answering communities act as collaboration networks of millions of users, which continue to generate huge amount of useful web content.

Currently, users of CQA sites post their questions and wait for other users to post answers to the question, which may even take several days. Even if someone posts the answers, the asker is sometimes not satisfied with the quality of the answer. In both the cases, a system that could recommend questions to suitable users is needed. The goal of such system is to link questions with experts who have higher likelihood of answering these questions. Such a system can contribute towards creation of high quality answers for questions and reducing the time of collecting high quality answer.

In this paper, we propose an approach to recommend lists of users, who can give high quality answers to new questions posted on CQA sites. Our main assumptions are: (1) best answerer would have high interest and expertise on the given question, (2) questions are generated from topics, (3) user expertise and interest on questions are affected by user expertise and interest on related topics. Based on the above assumptions, we collect historical answered questions, together with their answers, to generate profiles for each user. We apply the Latent Dirichlet Allocation (LDA) model on the user profiles to learn their interests on various topics. User topical expertise is modeled based on the voting information of historical answered questions. We make use of collaborative voting mechanism because it is supported by most of the CQA sites like Yahoo! Answers, StackOverflow, and etc. In addition, votes of an answer implies the answerer's expertise on related topics. We propose a model to represent the probability of a user to provide high quality answer to a particular question, by considering the user's interests and expertise on the topics of the question. The inputs to our approach are historical answered questions for each user and a new question, while the output is a list of users who are ranked by their probabilities of being the best answerer of the question. We have applied our approach to a corpus of questions and answers from StackOverflow, one of the biggest CQA sites. The results show that our approach outperforms the TF-IDF based approach.

The contributions of our paper are:

- We propose a new method to predict the best answerers for new questions on Community Question Answering (CQA) sites. Our approach considers both user interests and user expertise on topics of questions.
- Our approach learns user topical interest and expertise from historical answered questions, by leveraging topic modeling and collaborative voting mechanism.
- We compare our approach with a TF-IDF based approach on more than 99,000 questions extracted from StackOverflow, one of the biggest CQA sites. Our experiments show that our approach performs better than the TF-IDF based approach.

The structure of this paper is as follows. In Section 2, we describe previous related work. Next, we describe background information about StackOverflow and a baseline approach in Section 3. In Section 4, we introduce our approach for predicting best answerers for new questions. We analyze experiment results in Section 5. We conclude and mention future work in Section 6.

2 Related Work

Recommending the best answerer is a hot topic in the CQA research area and has garnered interest of many researchers. Several studies present algorithms to discover authorities in the communities [3,4]. Liu et al. applied information retrieval (IR) techniques to find experts on CQA site [3]. They computed a textual similarity between users' previously answered questions and new questions, and ranked the users according to the similarities. Zhang et al. leveraged network based algorithms such as PageRank and HITS to study network structure of Java Forum [4]. They proposed a method named ExpertiseRank based on a new authority evaluation metric Z-score and compared it with other network based recommendation methods. Qu et al. analyzed Yahoo! Answers and suggested that CQA sites should recommend questions to users who are interested in them [5]. They applied the Probabilistic Latent Semantic Analysis (PLSA) to model user interest. Liu et al. modeled answerers' behavior on Yahoo! Answers [6]. They investigated when and how answerers select and answer question. Different from above studies, we assume that questions are generated from topics, and we consider both user interest and user expertise on topic layer. We also leverage collaborative voting mechanism at CQA sites.

Our work is closely related to the following two studies. Liu et al. predicted the best answerer for new questions based on a model that combines language model and the Latent Dirichlet Allocation (LDA) Model [7]. They regarded users' reputation and activity as the prior probability of the user to answer a question. We follow their definition of user activity. However, instead of directly using the reputation of user as prior, we computer user expertise for each question. Riahi et al. recommended expert users through user profile collected from all the best answers given by users [8]. They tested different topic models, such as LDA and Segmented Topic Model (STM) on user profiles. However, they did not consider votes of questions and answers, as well as user activity.

3 Preliminary

3.1 StackOverflow

StackOverflow is one of the biggest question answering site where users can share knowledge, seek expert advice on a wide range of topics in computer programming. Users on StackOverflow have the ability to ask and answer questions, to vote questions up and down and several other features. StackOverflow employs gamification techniques to reward users for performing various set of actions.

Rewards include earning reputation points and badges, which when crosses the threshold, gives additional privileges to the users.

With more than 1.7 million users and over 4,000,000 questions, StackOverflow has become a huge knowledge repository. Each question is assigned tags according to the topic which question belongs to. The top six most discussed topics on the site are: C#, Java, PHP, JavaScript, Android and jQuery. Most of the questions are generally related to a specific programming problem, a software algorithm or software tools.

Each user has a reputation score, which signifies how much community trusts that user. Each question and answer can be voted up or down by other users who feel whether that question or answer is useful or not. Each question voted up fetches +5 for the user whereas each answer voted up increases the reputation of the user by 10 points. User loses reputation by 2 when an answer is voted down. Asker can accept one of the answers as the best answer, then the reputation of the best answer provider will increase by 15. Also, there is a limit on the votes that can be casted by a person in a day. Based on reputation points, users are given privileges like edit posts, retag questions, vote to close, reopen, or migrate any questions etc.

3.2 Baseline Approach Based on TF-IDF

Ranking potential answerers based on the similarities between their profiles and new questions is a basic approach to solve the best answerer prediction problem. The underlying idea is that, given a new question, the user who have answered similar questions should be recommended to answer the question. In this approach, user profiles and questions are stored as documents. TF-IDF based Vector Space Model is applied, where each document is represented as a vector of weighted features. Features are the words appearing in the document, and TF-IDF values are computed as the weights of features. Given one user content profile θ_u and a new question q, the approach ranks the users based on the cosine similarity between θ_u and q. The cosine similarity is denoted as

$$s(\theta_u, q) = \frac{\sum_w tfidf(w, \theta u) tfidf(w, q)}{\sqrt{tfidf(w, \theta u)^2}\sqrt{tfidf(w, q)^2}} \tag{1}$$

where w refers to words that appears in both user u's profile and question q. $tfidf(w, \theta u)$ is the tfidf weight of word w in θ_u. $tfidf(w, q)$ means the TF-IDF weight of word w in question q. Here, the TF-IDF weight of a word w in a given document d is defined as

$$tfidf(w, d) = \frac{\mathrm{f}(w, d)}{\max\{\mathrm{f}(w, d) : w \in d\}} \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{2}$$

where D denotes a collection of documents.

4 Approach

4.1 Overall Framework

In this section, we present the overall framework of our approach. The framework is illustrated in Figure 1. It contains four main parts: data preprocessing, user profile building, user topical interest & expertise learning, and ranking model building. Firstly, we generate user profiles from the previously answered questions. These profiles include both the textual information and voting information, which is then used to learn the topics as well as user interest and expertise on each learned topic. We propose a ranking model to compute the probability that a user becomes the best answerer of a given question. User interest and expertise on topics are captured in the ranking model. Finally, for each question in testing data, we rank the users based on the probability and find the Top-K users who have higher chances to provide the best answer. We introduce the major parts of our approach in details in the following subsections.

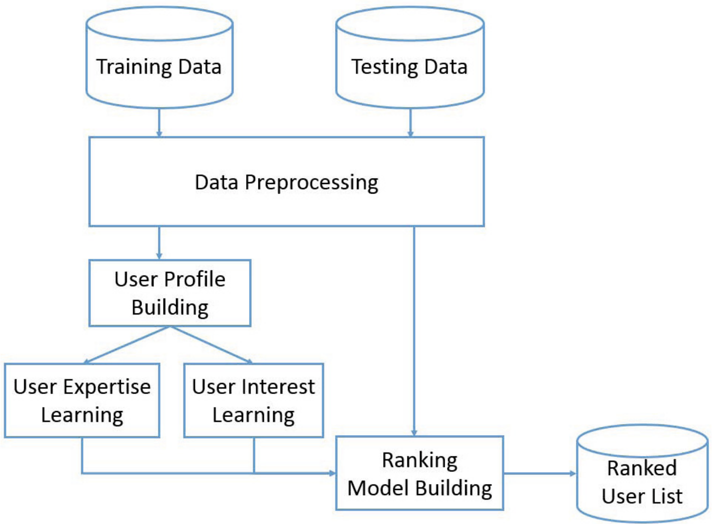

Fig. 1. Overall framework

4.2 Data Preprocessing

We crawl web pages of StackOverflow and collect questions and answers posted during year 2012 under tag “C#”, which has the most number of questions among all tags on StackOverflow[6]. Next, we extract both textual and code content for a post (question or answer) from the raw data and then implement the

[6] `http://stackoverflow.com/tags`

general text pre-processing steps: tokenization, stop-word removal, and stemming. Note that the content of a post includes code information, we add some special processes for code content to make the data clean, such as removing keywords in programming language, splitting function name, etc. Cleaned data is then separated into two parts: training dataset and testing dataset. Training data are used to learn user interest and user expertise on topics, while testing data are used to evaluate our approach. For training data, to avoid the impact of bias generated from especially high and low frequency tokens on our ranking model, we rank all the appeared tokens based on the term frequency. By default, tokens ranked in the top 1% and bottom 1% of the ranked list are removed from the corpus. Besides the textual and code information, voting information of answers are also recorded. For a particular answer, the voting information contains: votes and whether this answer has been accepted as the best answer by the asker.

4.3 User Profile Building

Each user profile contains all the answers provided by the user and their corresponding questions. To build these user profiles, we scan all the questions posted during a period of time, and find the users who have answered it. We then put the given answer, together with the question, to the profile of the answerer. In this work, we consider active users on StackOverflow as the potential answerers for new questions, based on the fact that only few of users on StackOverflow are responsible for a large number of questions [9,10]. This means only the profiles of the active users are used as training dataset.

4.4 Ranking Model

To predict the best answerer for a new question, we introduce a probability model. Formally, given a new question q, the probability of a user u being the best answerer for the question is defined as

$$P(u|q) = \frac{P(u)P(q|u)}{P(q)} \tag{3}$$

where $P(u)$ is a prior probability of user u to answer a question, and $P(q|u)$ denotes the probability of generating question q from user u's profile. $P(q)$ refers to a probability related with the question, which remains the same for all potential answerers. Therefore, we can rank answerers just based on the value of $P(u) \times P(q|u)$. Next, we describe the details to calculate $P(u)$ and $P(q|u)$.

Probability of Generating Question from User Profile. Intuitionally, user's answering behavior would be affected by his or her interest and expertise on the question. In other words, users who are interested in the question and also have the expertise related to the question are more likely to be the best answerer of the question. Based on this assumption, we compute the value of

$P(q|u)$ from two aspects: user interest and user expertise. The final probability model is computed as

$$P(q|u) = \prod_{w \in q} P(w|\theta_u)^{n(w,q)} \tag{4}$$

$$P(w|\theta_u) = (1 - \lambda)P_{interest}(w|\theta_u) + \lambda P_{expertise}(w|\theta_u) \tag{5}$$

Equation 4 is formed based on the assumption that each word in the new question q is generated independently. θ_u presents the profile of user u including both the content profile and the voting profile. $n(w, q)$ is the number of times that word w appears in question q.

Equation 5 states how we compute $P(w|\theta_u)$ by considering both user interest and user expertise. λ is a parameter to control the impact of these two aspects on the value of $P(w|\theta_u)$. In conclusion, our main tasks next are computing $P(u)$, $P_{interest}(w|\theta_u)$, and $P_{expertise}(w|\theta_u)$, respectively.

Prior Information of Answerers. In this work, we model $P(u)$, the prior probability of user u based on the activity of the user. Whenever a user posts a question, he or she wants the answer which is correct as well as timely. Some users could be active for a long time and they continue to give answers to newly posted questions whereas other ones answer occasionally. Therefore, we consider the activity of the user i.e., if a user gave an answer closer to the first question appearing in the test data, that user has a higher activity. In other words, a user who answered questions frequently in the beginning but lately gave very few or no answers would have a lower activity. Based on above assumption, we follow the work of Liu et al. [7] and define the user activity for a given question as

$$P(u) = Activity(u) = exp^{-(t_q - t_u)} \tag{6}$$

where t_q is the posting time of the new question in test data, t_u is the most recent time when the user authored an answer to a question.

4.5 Modeling User Topical Interest

A user will respond to particular question only if that question is related to the interest area of that user. So, we model user's interest based on the previous responses or answering history of that particular user as it shows the interest of the user on a particular topic. We use all the answers given by a particular user as well as the questions of all these answers to build up the user profile. The user profile has some latent topics within it.

One of the important problems in natural language processing is the lexical gap problem [11]. Language model used in various studies is based only on words matching and does not consider semantic information. So, it does not addresses the problem of lexical gaps between new questions and user profiles. Users have interests on several topics. In a typical context, user tries to choose the topic which he finds most interesting and then narrows down on the questions related to that topic. We use latent dirichlet allocation (LDA) model [12], which has been widely used in information retrieval. LDA has the ability to model topics in large corpus i.e., user profiles in our experiment. LDA model is represented as graphical model.

The topic mixture in LDA is drawn from a conjugate Dirichlet prior, which remains same for all the users. The process of generating user profile θ_u for a specific user u is shown in Figure 2 and described as follows: 1) ϕ_z is the multinomial distribution for each topic z from a Dirichlet distribution with parameter β. ϕ_z gives the word distribution within topic. 2) Pick a multinomial distribution θ_u for each user profile from Dirichlet distribution with parameter α. 3) Select a topic z $\in$ 1, ... K from the multinomial distribution θ_u for each word token w in user profile θ_u. 4) pick word w from the multinomial distribution ϕ_z. To generate the user profile, above procedure is repeated for N_u times where N_u is number of words in θ_u. Further, the above procedure is repeated N times for N users. The likelihood for the user profile collection is given as

$$P(u_1, ..., u_N|\alpha, \beta) = \prod_{z=1}^{K} P(\phi_z|\beta) \prod_{U=1}^{N} P(\theta_u|\alpha)(\prod_{i=1}^{N_u} \sum_{z_i=1}^{K} P(z_i|\theta)P(w_i|z_i, \phi))d\theta d\phi \tag{7}$$

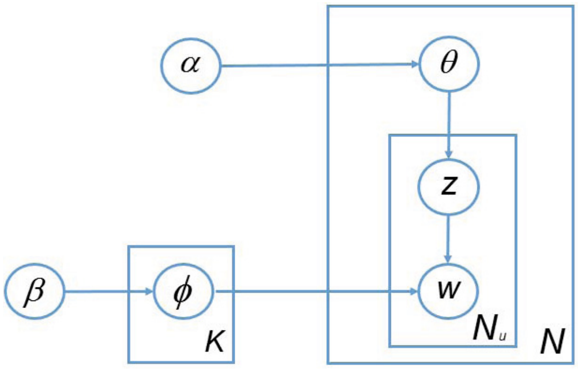

Fig. 2. User profile corpus generated by Latent Dirichlet Allocation

LDA presents a different representation compared to the language model for generating user profile based on user topics. LDA gives multiple topics for each user which shows that each user has varied interests. After we estimate θ and ϕ, the probability of generating a word from a user profile is given as

$$P_{LDA}(w|\hat{\theta}, \hat{\phi}, \theta_u) = \sum_{z=1}^{K} P(w|z, \hat{\phi})P(z|\hat{\theta}, \theta_u) \tag{8}$$

where $\hat{\theta}$ and $\hat{\phi}$ are the posterior estimates of θ and ϕ respectively.

As LDA model measures relations in the topic space rather than based on word matching, it does not require a word to appear in user profile to find correlations between a word and a specific user profile. Thus, it alleviates the lexical gap problem. Note that the value of $P_{LDA}(w|\hat{\theta}, \hat{\phi}, \theta_u)$ is used as $P_{interest}(w|\theta_u)$ in Equation 5.

4.6 Modeling User Topical Expertise

Given a new question, user who has more expertise on the question might have higher probability to be the best answerer. For each potential answerer, this expertise can be learned from his or her profile (content and voting). This expertise is defined as $P_{expertise}(w|\theta_u)$ in Equation 5.

As mentioned in Section 4.5, semantic similarity between the user profile and the new question can be captured through topic model. Therefore, we can compute $P_{expertise}(w|\theta_u)$ as

$$P_{expertise}(w|\theta_u) = P_{expertise} \sum_{z} P(w|z, \hat{\phi}) P(z|\theta u) \tag{9}$$

where $P(w|z, \hat{\phi})$ refers to a value irrelevant to user characteristic and $P(z|\theta u)$ refers to the expertise of user u on topic z. To keep the model simple, $P(w|z, \hat{\phi})$ is computed based on the model learned in Section 4.5.

We model user topical expertise by leveraging the collaborative voting mechanism. The major challenge is to map the user's expertise on a question to his or her expertise on topics. To address this challenge, we compute the topic distribution of each answer (corresponding question is also included) from the LDA model learned in Section 4.5. Next, we distribute the user's expertise on the question to his or her expertise on topics of the question. For instance, if one question is highly related to Topic z, then expertise value for topic z will be higher as compared to other topics. The more the user answers questions related to topic z, the higher the expertise of that user has on topic z. The pseudo code for measuring user topical expertise is shown in Figure 3.

For each user in the training dataset, we use a vector EV to store his or her expertise on each topic (Line 2). For each answer given by this user, function *getScore* computes the score of the answer (together with its question) based on the votes of the answer and its accepted state (Line 6). Function *getProbability* computes the topic distribution for the question using Tassign Matrix, an output of model learned in Section 4.5 (Line 8). Tassign Matrix contains the assigned topic for each word appearing in the user profile. For instance, $QA[i]$ contains 5 words, 3 of them are assigned to topic 1 while the other 2 are assigned to topic 2. Then the topic distributions of $QA[i]$ on topic 1 and 2 are 0.6 and 0.4, respectively. Next, we distribute the score of a question based on the probability of this question for each topic (Line 9-10). After scanning QA in the user profile, we achieve EV as the user expertise on each topic, this vector is then added into the final user-topic matrix EM (Line 11). In the end, we normalize all expertise by topics to make all the absolute values of topical expertise no more than 1 (Line 12). The output matrix EM is then used in Equation 9.

```
Procedure LearnUserExpertise
Inputs:
  $U$: User Profiles
  $VM$: Voting Matrix, contains voting information for each answer.
  $TM$: Tassign Matrix
  $K$: Topic Number
Output:
  $EM$: User Topical Expertise Matrix
Methods:
  1: For each user $u$ in $U$
  2:   Let $EV$ = a vector contains $u$'s expertise on each topic.
  3:   Let $QA$ = answers provided by $u$ and their corresponding questions.
  4:   For $(i = 1; i \leq |QA|; i++)$
  5:     Let $Score$ = the expertise of $u$ on the question $QA[i]$;
  6:     $Score = getScore(VM, u, QA[i])$
  7:     Let $Probability$ = a vector contains the topic distribution of question $QA[i]$
  8:     $Probability = getProbability(Tassign, u, QA[i])$
  9:     For $(j = 1; j \leq K; j++)$
  10:      Add $Probability[j] \times Score$ to $EV[j]$
  11:  Add $EV$ to $EM$
  12: $NormalizeByTopic(EM)$
  13: Output $EM$
```

Fig. 3. Learning user topical expertise

5 Experiment

5.1 Dataset

On StackOverflow, questions are assigned one or more tags, such as C#, Java, php, android etc. These tags are given by users to tell the category of a question. In this work, we create a dataset including questions and answers under tag C# that were posted between January 1, 2012 to December 31, 2012. We consider questions and answers under tag C# because currently C# has the highest number of questions tagged to it. Table 1 shows some basic statistics of our dataset.

Table 1. Data statistics

Tag	Questions	Askers	Answers
C#	99,166	37,363	191,338

We divide our dataset into two parts: training data and testing data. Training data include questions and answers posted from January 1, 2012 to November 30, 2012. Testing data contain questions posted from December 1, 2012 to December 31, 2012. We consider active users who have answered at least five questions during training data period as the potential answerers for testing questions.

We keep the testing questions that have an accepted answer and remove other testing questions. The answerer who gave the accepted answer to a testing question is regarded as the ground truth for that question.

5.2 Experiment Setup

User Interest Leaning. The input of the LDA model is a collection of user profiles. To learn topics' distributions on words and probabilities of user profiles generating topics, we use Gibbs sampling for inference and parameter estimation. By default, we set the number of topics as 100 and run LDA with 500 iterations of Gibbs sampling. Hyper parameters α and β are 0.5 and 0.1, respectively. Weight parameter λ is 0.5.

User Expertise Leaning. As shown in Figure 3, function $getScore$ is used to compute a value for an answer to represent its quality. In the experiment, we compute this value based on the voting rules on StackOverflow: each positive vote increases the value by 10, while each negative vote reduces the value by 2. If an answer has been marked as accepted answer, we add 15 to the value of the answer.

5.3 Research Questions

RQ1: Is our method effective in predicting the best answers for new questions compared to the baseline method ? To evaluate the performance of approaches, we follow the previous related paper. Approaches need to consider all the potential answerers and rank them according to their probabilities of being the best answerer. For each testing question, if the position of the best answerer is present in a range, say among the top N (N=1,50,or 100), of the returned ranked list, we call it a success at position N. We then compute a success rate S@N by dividing the success times by the number of total testing questions.

RQ2: What is the effect of varying the parameter λ ? Our approach takes both user interest effect and user expertise effect into consideration. The parameter λ is defined in Equation 5 to control the weights of these two effects. We vary this parameter to investigate how user interest effect and user expertise effect contribute to a better ranking model.

5.4 Experiment Results

RQ1: Effectiveness of Our Approach. We compare our approach to a TF-IDF based approach. We denote our approach that combines user interest and user expertise as $Our_{Combine}$. The parameter λ is 0.5. We also consider an approach $Our_{Interest}$ which only considers user interest, i.e., the parameter λ is 0. For the evaluation metric S@N, N is varied as 1, 3, 5, 10, 20, 50, and 100. Results of these three approaches are shown in Table 2.

Table 2. Comparison of S@N rate

Approach	**S@1**	**S@3**	**S@5**	**S@10**	**S@20**	**S@50**	**S@100**
TF-IDF based	0.20%	0.32%	0.45%	0.65%	1.09%	2.17%	3.42%
$Our_{Interest}$	1.18%	2.14%	2.66%	3.82%	5.53%	9.226%	13.76%
$Our_{Combine}$	2.11%	4.27%	5.48%	7.76%	11.11%	16.63%	21.50%

From the results shown in Table 2 , we find that our approach $Our_{Combine}$ performs best among the three approaches. With higher values of N, we observe significant improvement in $S@N$ ratio. This result shows that our approach is more efficient in predicting the best answerers than the TF-IDF based approach. $Our_{Interest}$ approach also performs better than TF-IDF approach as $Our_{Interest}$ considers the same content as baseline approach . This result suggests that topic model might capture more information than vector space model. We also note that $Our_{Combine}$ approach is better than $Our_{Interest}$ approach, which shows that the additional information about user expertise is useful.

RQ2: Effects of Varying Parameter λ. Parameter λ defined in Equation 5 controls the impact of user interest and expertise. We range the value of λ from 0.0 to 1.0: $\lambda = 0.0$ means only consider user interest and $\lambda = 1.0$ means only consider user topical expertise. The results with different λ values are shown in Table 3.

Table 3. Varying value of parameter λ

λ	**S@1**	**S@3**	**S@5**	**S@10**	**S@20**	**S@50**	**S@100**
0.0	1.18%	2.14%	2.66%	3.82%	5.53%	9.226%	13.76%
0.3	2.08%	3.65%	4.74%	6.84%	9.53%	14.28%	18.92%
0.5	2.11%	4.27%	5.48%	7.76%	11.11%	16.63%	21.50%
0.7	2.14%	**4.35%**	**5.71%**	**8.38%**	11.86%	18.18%	**23.06%**
1.0	**2.18%**	4.15%	5.56%	8.33%	**12.57%**	**18.34%**	23.01%

Table 3 shows that as the value of λ increases, the success rate generally increases. The numbers in the bold highlight the highest success rate under each measurement. The setting with $\lambda = 0.7$ has the highest values for 4 out of 7 measurements, while $\lambda = 1.0$ has the highest values for the other three measurements.

5.5 Threats to Validity

Threats to internal validity refers to the representative of our model. The empirical selection of parameter like α, β, topic number K, Gibbs sampling iteration n might influence the results of LDA. We measure the quality of a question based on its votes, and the reputation rules on StackOverflow. However although the votes are given by the community, there might be human bias. For instance, user might prefer to vote up answers with more votes, or answers that are accepted, or answers given by experts.

External validity is related to the generalizability of our results. Our dataset consists of more than 99,000 questions posted from January 1, 2012 to December 31, 2012 which may not represent all the types of questions asked under category C#. Also, since we consider only questions marked under C# tag, the results may not hold true for questions belonging to other category.

Threats to construct validity corresponds to the appropriateness of the evaluation criteria. Following other similar works, we regard the user who provides the accepted answer for the test question as the ground truth based on the assumption that the answer marked as accepted has the highest equality among others. But the accepted answer is labeled by the asker, which might not be the best answer that selected by the whole community.

6 Conclusion

In this paper, we proposed an approach that combines user interest effect and user expertise effect on topic layer, to predict best answerers for new questions on Community Question Answering (CQA) sites. By using historical answered questions of users, we model the interests of the answerers using Latent Dirichlet Allocation (LDA). We also incorporate collaborative voting mechanism at CQA sites to learn user expertise on each topics. We compare the performance of our approach with the TF-IDF based approach on a dataset extracted from StackOveflow. The results show that our approach can improve the effectiveness of the baseline approach.

As a future work, we intend to expand our study to include more questions from various community question answering sites. We also plan to investigate whether we can persuade those users who do not participate actively by recommending them questions that they are interested in and have enough expertise to solve. This would mitigate the problem of low participation rate commonly faced by community question answering sites.

References

1. Liu, Y., Bian, J., Agichtein, E.: Predicting information seeker satisfaction in community question answering. In: SIGIR, pp. 483–490 (2008)
2. Liu, Q., Agichtein, E., Dror, G., Gabrilovich, E., Maarek, Y., Pelleg, D., Szpektor, I.: Predicting web searcher satisfaction with existing community-based answers. In: SIGIR, pp. 415–424 (2011)
3. Liu, X., Croft, W.B., Koll, M.B.: Finding experts in community-based question-answering services. In: CIKM, pp. 315–316 (2005)
4. Bouguessa, M., Dumoulin, B., Wang, S.: Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In: KDD, pp. 866–874 (2008)

5. Qu, M., Qiu, G., He, X., Zhang, C., Wu, H., Bu, J., Chen, C.: Probabilistic question recommendation for question answering communities. In: WWW, pp. 1229–1230 (2009)
6. Liu, Q., Agichtein, E.: Modeling answerer behavior in collaborative question answering systems. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 67–79. Springer, Heidelberg (2011)
7. Liu, M., Liu, Y., Yang, Q.: Predicting best answerers for new questions in community question answering. In: Chen, L., Tang, C., Yang, J., Gao, Y. (eds.) WAIM 2010. LNCS, vol. 6184, pp. 127–138. Springer, Heidelberg (2010)
8. Riahi, F., Zolaktaf, Z., Shafiei, M., Milios, E.: Finding expert users in community question answering. In: Proceedings of the 21st International Conference Companion on World Wide Web, pp. 791–798. ACM (2012)
9. Wang, S., Lo, D., Jiang, L.: An empirical study on developer interactions in stackoverflow. In: 28th ACM Symposium on Applied Computing (2013)
10. Xia, X., Lo, D., Wang, X., Zhou, B.: Tag recommendation in software information sites. In: Proceedings of the Tenth International Workshop on Mining Software Repositories, pp. 287–296. IEEE Press (2013)
11. Berger, A., Caruana, R., Cohn, D., Freitag, D., Mittal, V.: Bridging the lexical chasm: Statistical approaches to answer-finding. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2000)
12. Blei, D.M., Ng, A.Y., Jordan, M.I., Lafferty, J.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 2003 (2003)