Recommending Answerers for Stack Overflow with LDA Model

Bin Shao
State Key Lab of Software Development Environment
School of Computer Science and Engineering, Beihang University
Beijing, China
shaobin@act.buaa.edu.cn

Jiafei Yan
State Key Lab of Software Development Environment
School of Computer Science and Engineering, Beihang University
Beijing, China
yanjf@act.buaa.edu.cn

ABSTRACT

Stack Overflow is the largest Community-based Question Answering (CQA) site for software developers. Its popularity is mainly attributed to the timely answers provided by a great number of developers. When having problems in learning and using new technologies, developers resort to Stack Overflow for help. However, it is difficult to recommend questions to the potential answerers due to the huge numbers of questions. In order to improve the accuracy of question recommending, we need to find out the members who are interested in the fields related to the questions and match the ability of developers with the difficulty of questions. To do so, we need to pay close attention to the behavior of developers. This paper presents a model with two prediction approaches, namely, the traditional feature-based approach and LDA (Latent Dirichlet Allocation) based approach. When a new question arrives, this model will use LDA method to label the question and indicate the proper category to which the question belongs according to latent semantic feature and content feature. Then, with the traditional features of the question and the asker information, the model will recommend the appropriate developers to answer this new question.

CCS CONCEPTS

• **CInformation systems → Recommender systems**;

KEYWORDS

Stack Overflow, LDA, classifier, recommender system

ACM Reference format:

B. Shao, and J. Yan. 2017. Recommending Answerers for Stack Overflow with LDA Model. In *Proceedings of ChineseCSCW '17, Chongqing, China, September 22-23, 2017*, 7 pages.
https://doi.org/10.1145/3127404.3127426

1 INTRODUCTION

With the coming age of big data and the vigorous development of Internet, question and answer (Q&A) sites have become one of the important methods for data retrieving and obtaining. Community-based Question Answering (CQA) sites, like Yahoo! Answers and Quora, have become the most popular platforms for people to solve the problems in daily life. In particular, Stack Overflow is one such site for programmers and software engineers.Since launched by Jeff Atwood and Joel Spolsky in 2008, Stack Overflow has grown into the largest CQA site, where the programmers and software engineers can look for the knowledge and solutions related to computer and programming problems, the community members can ask, answer or participate in discussions. Stack Overflow design incentive mechanisms with reputation and badges to motivate community members to answer questions, vote on questions and answers, and participate in discussions. Community members can improve impacts by earning reputation and badges. The higher the impacts, the more gain. Every question or answer in Stack Overflow has its score, which is made by voting (i.e. up vote, down vote). The voting score serves as a rough measure of Q&A quality. Through these measures, Stack Overflow attracts more and more users. As of September 2016, Stack Overflow had nearly 6 million registered users, more than 12.3 million questions, and more than 19.7 million answers.

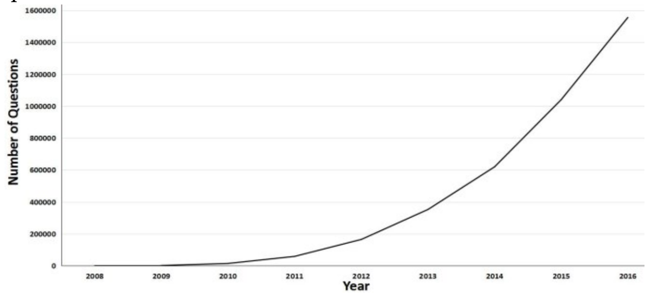

Figure 1: **Number of questions without answers**

Although people got involved in answering are increasing, the number of unanswered questions is increasing even more rapidly. On the one hand, the large amount of questions makes it very difficult for community members to search and find questions that they are interested in, on the other hand, even though community members discovered the interesting questions, they may not be able to solve them. Fig. 1. shows the number of questions that remain unanswered each month (we considered Stack Overflow data from September 1, 2008 to September 1, 2016). As illustrated in the figure, the amount of unanswered questions is growing faster and faster, and there is a rapid growth in unanswered questions over the last years. In order to alleviate this situation, Stack Overflow recommends questions to community members on the basis of the tags, which are collected or from the questions users once answered.

Furthermore, the proportion of questions without any answer has been increasing every year (see Table 1). To reduce the number of the unanswered questions and help community members to solve the problem, it will be useful to recommend a question to appropriate users who are interested in answering and have the ability to answer. This problem is known as question routing in CQA, which is the focus of this paper. We predict who will answer a question based on the features of the question itself and questioner who ask this question. Our prediction is made when the question is created, so, it can be used as a recommending method.

Table 1: Proportion of Unanswered Questions by Year

Year	Summation of questions	Number of unanswered questions	[%]
2008	58,844	65	0.11
2009	404,529	1,757	0.43
2010	1,107,223	14,811	1.34
2011	2,319,811	59,703	2.55
2012	3,982,819	165,084	4.14
2013	6,057,004	352,637	5.82
2014	8,233,921	619,869	7.53
2015	10,556,922	1,040,679	9.86
2016	12,350,340	1,555,324	12.60

Towards the goal of recommending the appropriate developers to answer this new question, we first cluster the questions by using LDA, this method allows us to partition the questions into smaller sets than using tags. We then obtain features from four different views: 1.) the topic characteristic by LDA, 2.) the traditional features of questions (i.e. length, the code fragment, link, etc.), 3.) the characteristic of the time and 4.) the features of a user (i.e. the length of membership, reputation, age, the number of accepted answers, etc.). We then conduct experiments to understand their features and find the most similar questions by cosine similarity. On the basis of previous work, we create a classifier to distribute the question to the appropriate developers to answer. In general, we make the following research contributions:

We analyze questions on Stack Overflow posted over 7 years and conduct a topic study. To make a more detailed classification of the questions, we discard tags which are supplied to the questions by Stack Overflow, and cluster questions through LDA. This provides the basis for relevant research.

We develop a predictive model using some machine learning frameworks to recommend the appropriate developers to answer the questions. We experiment with 18 features on posts set which is labeled as Java on Stack Overflow. Our prediction approach achieved 52% F-measure.

2 Related work

In this section, we briefly describe previous work related to our study.

2.1 Stack Overflow

Stack Overflow a part of Stack Exchange website, it is a Community-based Question Answering site for programmers, software developers and some other people who have problems with programming. The site was created in 2008 by two programmers, Joel Spolsky and Jeff Atwood. Stack Overflow is designed for coders and computer geeks to ask, answer and discuss questions about programming [27]. With more and more developers and software engineers taking part in, Stack Overflow is becoming communities that community members discuss and solve problems. Meanwhile, the popularity of Stack Overflow inspired spin-offs to support community demands in other topical areas, collectively referred to as the Stack Exchange. Stack Exchange hosts a large network of question and answer sites (169 and growing), on diverse topics from software programming to game to photography and cooking. Stack Exchange covers many fields and there is no limit for people to use the sites.

In order to attract users to register and contribute to Stack Overflow rather than just posting questions and seeking answers. Stack Overflow devised a reputation and badge-earning system. The reputation, like the score of members, show the contribution the members made. It reflects the involvement of a member and the quality of their questions and answers.And the badges, like a task system of the game, encourage members to complete appropriate activities. For Example, when you have answered specified number of Java questions, you will get a badge about successfully answered questions about Java.

Due to the use of a large number of community members, Stack Overflow has a huge amount of data about programming and developers, and the operator of Stack Overflow make their data publicly available for research purposes termly. Therefore, many related studies on these data sets can be done. Some researchers have studied about the questions. Mamykina et al. and Asaduzzaman et al. did a statistical analysis of the average time taken for a question to be answered, and made a study to find out the reason why some questions were not answered for a long time and then gave community members some suggestions on arising questions [1–2]. Ahasanuzzaman et al. and Xu et al. did some research to find out the duplicate questions, to avoid duplicate questions [2–3]. Yao et al. and Baltadzhieva et al. studied about the quality of the questions, they think of the score of questions as the quality evaluation result arise from crowdsourcing, they also gave community members some suggestions to improve the quality of questions [4–5]. Similarly, Correa et al. studied about the deleted questions to warn community members not to ask low-quality questions [6].

Some other researchers pay close attention to the data beyond questions. Beyer et al. focus his attention on the tags from Stack Overflow, he studied to find out the similar tags and merge them to avoid the flood of tags [7-8]. Meanwhile, Halavais et al. showed solicitude for badges and has a study about the excitation mechanism [9]. Pinto et al. looked for developers' interest in software power issues [20]. Vasilescu et al. tried to establish an association of members between Stack Overflow and Github [10]. Wang et al. studied about the interaction between community members [11]. Lin B focus his attention on the gender of the developers [12]. Bazelli et al. used five personality models to study the personality features of different groups who use different technology [13]. Bajaj et al. studied trends in hot topics of the Internet by use topic models [21]. Gantayat et al. paid close attention to the difference between the accepted answer and the answer which has the highest votes [22]. Zagalsky et al. used Stack Overflow data of answers to design a code search engine for APIs [23]. Subramanian et al. used Stack Overflow data to complete some cases for the API documentations, to help developers who read these documentations [24].

2.2 Question Routing in CQA

In this paper, we study a routing question for collaborative answering in community question answering site. We aim to find out the regular pattern of community members answering questions and to route questions to the potential community members to answer.

Chang et al. proposed methods to construct co-occurrence graphs from asking, answering, commenting, voting, tagging and community members' availability data, and then to predict who would be willing to collaborate to answer that question [26]. Gou et al. used Bayesian networks to model a dependency between User-Question-Answer, and then to recommend a given question to potential community members [27]. Zhang et al. and Jurczyk et al. collected questions and answers data to build a collaboration network of community members, and use this network to recommend questions to potential answerers [28–29]. Zhou et al. used feature-based classification for question routing, yet because of the huge numbers of question-user pairs to be classified, their approach might be inefficient [30]. Choetkiertikul M et al. presented two approaches namely feature-based and social network based on route questions [32]. Hu et al. utilized classification techniques based on historical records to recommending developers to assign to a bug report during the triaging process [31].

2.3 LDA

LDA (Latent Dirichlet Allocation) is a generative statistical model of natural language processing. It can be used to generate topics for documents. We can do many things with topics. Zhao et al. used LDA to analysis emotions [33]. Titov et al. used LDA on social media site [34]. LDA is a very practical and effective method to process text data and extract latent semantic features.

LDA is an unsupervised method, and it is difficult to set up the number of topics. In order to evaluate LDA quantitatively, we treat it as a language model. We can use perplexity and the average similarity between each topic to evaluate LDA. These two evaluations will be converged when the number of topics changes. We consider the model is optimal when the evaluations convergence.

3 METHODOLOGY

3.1 Data Set

We download the Stack Overflow data dump from Stack Exchange. This data includes the publicly available history of questions and answers posts, tags, votes, users information and comments from August 2008 to September 2016. We mainly use the questions, answers and user information as data, and the rest data are used as supplementary information.

3.2 Preprocessing

The data dump we downloaded is in XML format, we convert the data and store it in MySQL. Before the experiment, we preprocess the initial data. Preprocessing consists of two aspects: 1.) On the one hand, we use the natural language processing (NLP) method to process the text data (questions' title, body and answers' body), including taking the word, stem and remove the stop words. We use these processed data for LDA. 2.) On the other hand, we use statistical methods to extract the traditional features of questions, answers, and users.

The preprocessed data saved as the basis for LDA and user recommendations, and provided features for the follow-up experiments.

3.3 LDA

In this paper, we use LDA to obtain the topic features of the questions and to cluster the questions by their topics. Each cluster has its own unique numbers as its label.

How many numbers of the topic should be set up? In the other words, how many clusters should be set up that we can get the best results? By comparing perplexity and the average similarity, for evaluation to find out the best number of topics. We use the Cosine Similarity to measure the similarity between the two vectors of the topics:

$$\text{similarity}(\mathrm{t}_i, \mathrm{t}_j) = \frac{\sum_k^n t_{ik} \times t_{jk}}{\sqrt{\sum_k^n (t_{ik})^2} \times \sqrt{\sum_k^n (t_{jk})^2}} \quad (1)$$

Where t_i and t_j are the two different topics, and t_{ik} and t_{jk} are the ith feature of topic t_i and t_j respectively, and n is the number of features, and n is the number of words. As shown in Fig. 2., we select 30 as the number of topics.

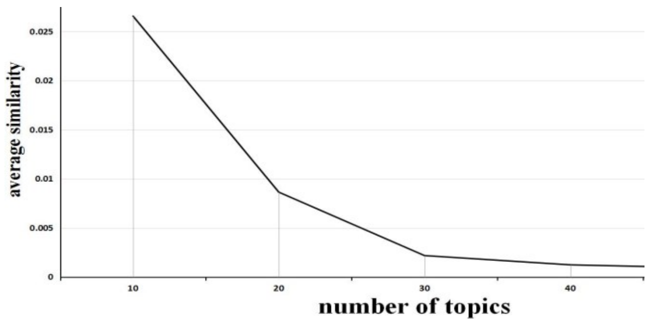

Figure 2: Correlation between topics and average similarity

3.4 Extracted Features

The extracted features of a question are applied in two aspects: 1.) finding the similar questions and clustering them together and 2.) using the features of questions and user information to train the classifier. For the first aspect, we use LDA to process the previous data which have been preprocessed by the NLP method to extract topic, and then, we use the topics to cluster the similar questions together. The idea is to divide the new questions into the clusters of similar questions, and thus make it easier to find out the potential answerers from community members. For the second aspect, we aim to extract the traditional features of the new questions and the users who asked this question. We can extract the structure of the question and the base attribute of the community member who asked the new questions. Each question is labeled a number in the first place. The number represents the cluster to which the question belongs. Then each question and its owner are encoded into a feature vector, which is used for the classifier to predict the community members as the potential members to answer the new question.

1) Topic feature: We use LDA to cluster questions, and the new question will be divided into one cluster. Every question in the same cluster have the same topic, and hence, has similar content.

2) Question features: We extract not only the features of the structure of the question, but also the features about the time of creation and readability:

- Create time in the day: We compute the time periods when the question arrived by using the CreateDate in data. In different periods of a day, online users are different.
- Create time in the week: We compute the day in a week when the question arrived. Users' activities are generally different in a week.
- Length of title: We measure the length of the title. Long title and short title may draw an attention from different users.
- Length of body: We measure the length of the body part of a question. If the body part is very lengthy, it means the question might have been described in rich details. If the body part is rather short, it means the question might be written in the simple and clear language. Different users may have different preferences on the length of question body.
- Number of code fragments: counted the number of code fragments provided in a question. All the questions in Stack Overflow are problems about programming or computer, moderate code fragments could make it easier for other users to understand the problem described in the question.
- Number of external links: counted the number of external links provided in a question. External links might provide extra information related to the question.
- Number of images: counted the number of images provided in a question. Some developers are accustomed to putting the screenshot in the question instead of copy the problem into the question. Therefore, the image might also provide more information related to the question.
- Readability score: measured the readability score of a question. There are six types of readability score: SMOG, Flesch Reading Ease, Flesch-Kincaid Grade Level, Automated Readability Index, Coleman-Liau Index and Gunning Fog Index. First, we computed the six readability scores as the features. Then we use feature filter of Weka to filter the features. Finally, we selected Automated Readability Index and Coleman-Liau Index as the readability score for our data.

3) Features of user: We extracted eight features from users:

- Reputation: measured how much a user helped others. Reputation score goes up when others vote "up" for the questions, answers and edits, and goes down when others vote "down" for the questions, answers and edits.
- Length of membership: measured the length of membership of the users. Since we have downloaded the full data from August 2008 to September 2016, we subtracted the creating time of user accounts from the creating time of the questions.
- Age: If the user had not registered his/her age, we fill age with the average of all valid age fields.
- Up Votes: counted the "up" voting score of a user.
- Down Votes: counted the "down" voting score of a user.
- Number of questions: counted the questions asked by a user.
- Number of answers: counted the answers of a user.
- Number of accepted answers: We count the accepted answers of a user.

3.5 Approach

Because of the approach is supervised, the approach can be separated into two partitions: training partition and predicting partition. Fig. 3. shows the detailed framework of our approach.

3.5.1 Training partition. In the training partition, we train two models, LDA and Random Forest. The first step is to automatically train the topic medal LDA by using the question text data, which have been preprocessed by using the NLP method on the data of questions title and body. Using LDA, we cluster the questions and number each cluster. Certainly, the topic model have been trained successfully.

The second step is, we train the Random Forest on every cluster. We use nine features of questions and eight features of the user who asked the corresponding questions to build the vector, and use the answerer ID as the label of this vector. Because one question may have more than one answer, this would lead to one vector having different labels. To solve this problem, a tiny deviation is added to the time variable, the deviation is too small that we can consider these two-time variables represent the same time. We can obtain a lot of question-user pairs, in each pair, if the user answer the question, the pair will be regarded as this user's class. For example, firstly, we have a cluster of questions on the same topic, there are m questions in the set, recorded as q_1, q_2 q_3 ... q_m. Then we can easily get a set of users who have answered at least one question in this cluster. There are n users in the set, recorded as u_1, u_2 ... u_n. Then we combine questions and users in each cluster by the relationship between questions and users, like (q_1, u_1), (q_1, u_2), (q_2, u_2), (q_2, u_3) ... (q_m, u_n). (q_1, u_1) representing that u_1 answered the question q_1, and so on. Here we have two users u_1 and u_2 both answered the question q_1, and we have user u_2 answered two questions q_1 and q_2. Lastly, we feed these pairs into the Random Forest to train the classifier.

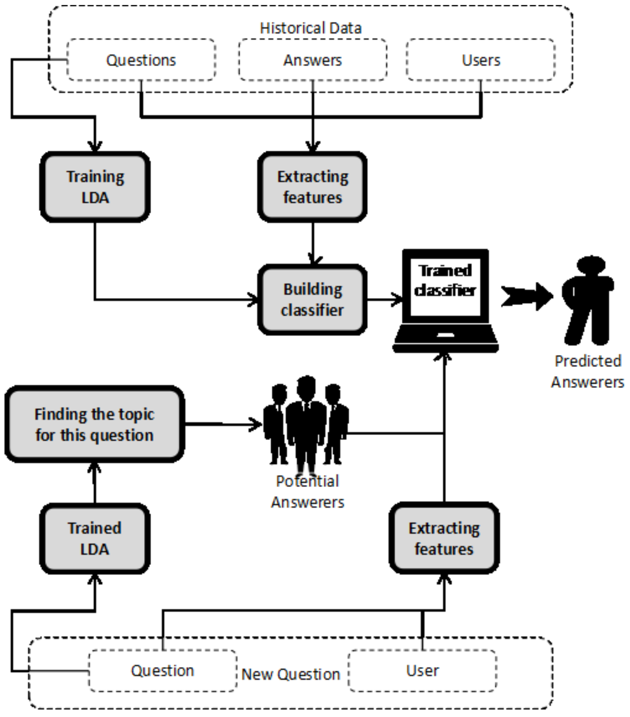

Figure 3: Prediction approach

3.5.2 Predicting partition. In the predicting partition, we have mainly two steps. The first step, we feed the new question into the LDA model that have been trained. The new question will get a topic number which representing the cluster which this question is divided into. This process can be seen as finding out the similar questions with this new question. Thus, we will get a set of potential answerers through these questions, and the set of the answerer ID of the potential answerers will be used as the label in the second step.

The second step, we build the vector by nine features of questions and eight features of the user who asked this question. Then we feed the vector into the trained classifier Random Forest to get the predicted answerers. We do not take the result of the classifier directly. We obtain the distribution of the results and sorted them. At last, we recommend answerers according to the sorted results.

4 Evaluation

We divide our data into a training set and a test set. The data are sorted by time stamp, we select the first 80% data as the training set, and the remaining 20% as the test set.

The effectiveness of our approach is reflected through the recommendation effect achieved by the predictive model. For each question in the test set, we collect the information of users who have answered this question as the result set. If the predicted user is included in the result, we consider this a true positive (tp). We then compute Accuracy of the prediction results:

Accuracy: the ratio of the quantity of the true positive to the total amount of the real results, which is the set of users who have answered this question. It is calculated as:

$$\text{Accuracy} = \frac{tp}{num_{result}} \tag{2}$$

num_{result} is the total amount of the real results.

Since there are only two to five answers for the most of the questions (by statistics, we can obtain that there are nearly 93% questions only have two to five answers), we decide to use the average accuracy to evaluate our method.

Table 2 shows the results of our evaluation for each topic. We present the results for accuracy for top-5, accuracy for top-10 and accuracy for top-20. In the table, the accuracy for top-5, top-10 and top-20 are the average value of the questions belong to the three topics. From the table, we can see that when we recommend 10 users, in the most of the topics, we will get the right one who had really answered this question. When we recommend 20 users, there is at least one right user who answered the question.

In order to understand what lead to the result, we perform a statistical analysis of the data set. Fig. 4. shows, we can easily find that only around 10% questions in our data have more than 3 answers. And in Fig. 5., we can see the numbers of questions in every topic, we can also easily find that most questions have only one or two answers. Hence, when we recommend the real answerer for the question, the accuracy raised by 30% to 50% immediately.

In addition, as Fig. 5. shows, there are obvious differences in the number and distribution of the questions under different topics. This will affect the quantity and quality of the data of training set, in the same way, this will also affect the classification effect of the Random Forests. So we can see that the accuracy is different under different topics.

Table 2: Evaluation Results

Topic Index	Accuracy (%)		
	Top-5	Top-10	Top-20
1	90.40%	91.59%	97.74%
2	0.00%	97.00%	97.50%
3	0.00%	0.00%	96.07%
4	0.00%	95.89%	95.78%
5	0.00%	93.15%	95.09%
6	0.00%	0.00%	92.63%
7	0.00%	0.00%	91.21%
8	0.00%	0.00%	90.55%
9	31.76%	31.76%	61.55%
10	0.00%	49.49%	49.96%
11	0.00%	49.48%	49.65%
12	46.42%	46.99%	49.31%
13	0.00%	0.00%	49.27%
14	0.00%	45.46%	49.11%
15	47.93%	47.87%	48.30%
16	0.00%	0.00%	48.20%
17	0.00%	47.19%	47.82%
18	0.00%	0.00%	47.66%
19	0.00%	23.41%	47.47%
20	0.00%	0.00%	47.29%
21	0.00%	0.00%	47.24%
22	0.00%	0.00%	47.06%
23	0.00%	0.00%	46.53%
24	0.00%	0.00%	46.02%
25	0.00%	0.00%	45.93%
26	0.00%	30.26%	33.15%
27	0.00%	23.36%	23.87%
28	23.46%	23.59%	23.65%
29	0.00%	18.30%	18.66%
30	0.00%	8.67%	8.80%

In order to show the advantage of this method, we preprocess our data with the user based collaborative filtering method. Although we can obtain some recommended results by adjusting the numbers of neighborhoods and recommenders, there was no right answer when compared with real data. This is because of the fundamental difference between questions and commodities. Given that two customers had bought many similar commodities before, when one of them bought some new items, we can assume that these new items are also preferred by the other customer, and thus recommend to users. But for questions, even if two users have answered many similar questions before, we can not assume that they will still answer similar questions in the future. That's because they might not be able to answer same questions. Therefore, we can see why the user based collaborative filtering approach do not work well in recommending respondents to software development questions.

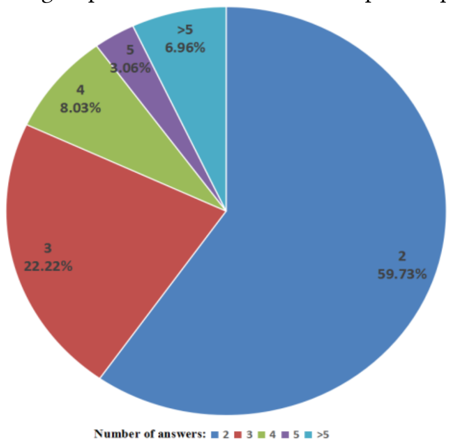

Figure 4: Distribution of questions

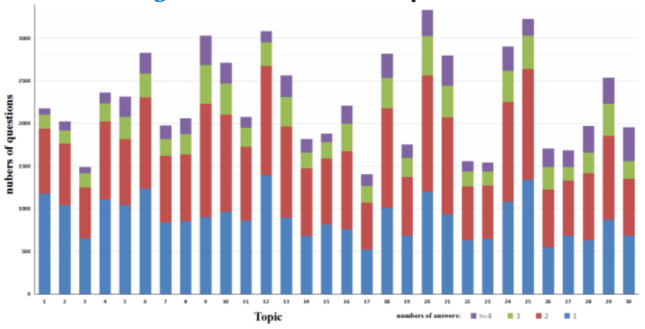

Figure 5: Numbers of questions

5 Threats to validity

External Validity: the threat to external validity refers to the data we selected. In this study, we select the questions with "Java" tag. Still in the future, we would like to extend this study to include more questions with other tags, like python, MySQL and so on.

Internal Validity: the threat to internal validity of our result mainly refers to experimenter biases. In this study, the most process is automated except LDA, we choose the number of topics through computing the average cosine similarity between different topics. However, this choice may not be applicable for training the classify model.

Construct Validity: we have designed a new vector to store the features of the questions and the askers. We solve the multi-label problem by adding a tiny deviation on the time, this method may cause some training set to be too large. Accordingly, we would like to try some different method to solve the multi-label problem.

6 Conclusion and Future work

The major challenge of Community-based Question Answering (CQA) sites is the difficulty in finding the appropriate users to answer the questions, this is especially true of Stack Overflow. The skills of developers are increasing in diversity with the emergence of new technology, how to recommend the answerers for the questions about programming is an important challenge. In this work, we use LDA to cluster the questions by their latent context features, which allow quick access to the similar questions when a new question arrived. We employ Random Forests classifier to recommend the users, and we can obtain a good result when we recommend 10 or 20 users.

In the future study, on the one hand, we find out that there are a lot of users who have answered only one question, this has a negative effect on our recommendations. How to reduce this negative effect is one aspect of the following work. On the other hand, we know there is a large number of new users signed up for Stack Overflow every day, and a part of whom may like to answer questions. How to recommend these users to the questions is an import future direction of our work. On other aspects, we will try certain methods to select the features and set the hyper parameter to enhance the effect of the recommending method. And we will also focus on the relationship between the distribution of data and categorization effectiveness, so as to improve categorization.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China under Grant No.2016YFB1000804.

REFERENCES

[1] Asaduzzaman M, Mashiyat A S, Roy C K, et al. Answering questions about unanswered questions of stack overflow[C]Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013: 97-100.

[2] Ahasanuzzaman M, Roy C K, et al. Mining duplicate questions in stack overflow[C]Proceedings of the 13th International Workshop on Mining Software Repositories. ACM, 2016: 402-412.

[3] Xu B, Ye D, Xing Z, et al. Predicting semantically linkable knowledge in developer online forums via convolutional neural network[C]Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ACM, 2016: 51-62.

[4] Yao Y, Tong H, Xie T, et al. Detecting high-quality posts in community question answering sites[J]. Information Sciences, 2015, 302: 70-82.

[5] Baltadzhieva A, Chrupała G. Question Quality in Community Question Answering Forums: a survey[J]. ACM SIGKDD Explorations Newsletter, 2015, 17(1): 8-13.

[6] Correa D, Sureka A. Chaff from the wheat: characterization and modeling of deleted questions on stack overflow[C]Proceedings of the 23rd international conference on World wide web. ACM, 2014: 631-642.

[7] Beyer S, Pinzger M. Synonym suggestion for tags on stack overflow[C]Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension. IEEE Press, 2015: 94-103.

[8] Beyer S, Pinzger M. Grouping android tag synonyms on stack overflow[C]Proceedings of the 13th International Workshop on Mining Software Repositories. ACM, 2016: 430-440.

[9] Halavais A, Kwon K H, Havener S, et al. Badges of friendship: social influence and badge acquisition on Stack Overflow[C]2014 47th Hawaii International Conference on System Sciences. IEEE, 2014: 1607-1615.

[10] Vasilescu B, Filkov V, Serebrenik A. StackOverflow and GitHub: Associations between software development and crowdsourced knowledge[C]Social Computing (SocialCom), 2013 International Conference on. IEEE, 2013: 188-195.

[11] Wang S, Lo D, Jiang L. An empirical study on developer interactions in stackoverflow[C]Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013: 1019-1024.

[12] Lin B, Serebrenik A. Recognizing gender of stack overflow users[C]Proceedings of the 13th International Workshop on Mining Software Repositories. ACM, 2016: 425-429.

[13] Bazelli B, Hindle A, Stroulia E. On the Personality Traits of StackOverflow Users[C]ICSM. 2013: 460-463.

[14] Chang S, Pal A. Routing questions for collaborative answering in community question answering[C]Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. ACM, 2013: 494-501.

[15] Guo J, Xu S, Bao S, et al. Tapping on the potential of q&a community by recommending answer providers[C]Proceedings of the 17th ACM conference on Information and knowledge management. ACM, 2008: 921-930.

[16] Jurczyk P, Agichtein E. Discovering authorities in question answer communities by using link analysis[C]Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. ACM, 2007: 919-922.

[17] Zhou T C, Lyu M R, King I. A classification-based approach to question routing in community question answering[C]Proceedings of the 21st International Conference on World Wide Web. ACM, 2012: 783-790.

[18] Hanrahan B V, Convertino G, Nelson L. Modeling problem difficulty and expertise in stackoverflow[C]//Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion. ACM, 2012: 91-94.

[19] Movshovitz-Attias D, Movshovitz-Attias Y, Steenkiste P, et al. Analysis of the reputation system and user contributions on a question answering website: Stackoverflow[C]//Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on. IEEE, 2013: 886-893.

[20] Pinto G, Castor F, Liu Y D. Mining questions about software energy consumption[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014: 22-31.

[21] Bajaj K, Pattabiraman K, Mesbah A. Mining questions asked by web developers[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014: 112-121.

[22] Gantayat N, Dhoolia P, Padhye R, et al. The synergy between voting and acceptance of answers on stackoverflow, or the lack thereof[C]//Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015: 406-409.

[23] Zagalsky A, Barzilay O, Yehudai A. Example overflow: Using social media for code recommendation[C]//Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering. IEEE Press, 2012: 38-42.

[24] Subramanian S, Inozemtseva L, Holmes R. Live API documentation[C]//Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 643-652.

[25] Atwood J, "Stackoverflow careers: Amplifying your awesome," Code Horror, 2009, http://www.codinghorror.com/blog/2009/11/stack-overflow-careers-amplifying-your-awesome.html

[26] Chang S, Pal A. Routing questions for collaborative answering in Community Question Answering[C]// Ieee/acm International Conference on Advances in Social Networks Analysis and Mining. IEEE, 2013:494-501.

[27] J. Guo, S. Xu, S. Bao, and Y. Yu, "Tapping on the potential of q&a community by recommending answer providers," Proceeding of the 17^{th} ACM conference on Information and knowledge mining - CIKM '08, pp. 921–930, 2008.

[28] Guo J, Xu S, Bao S, et al. Tapping on the potential of q&a community by recommending answer providers[C]// ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, Usa, October. DBLP, 2008:921-930.

[29] Jurczyk P, Agichtein E. Discovering authorities in question answer communities by using link analysis[C]// Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November. DBLP, 2007:919-922.

[30] Zhou T C, Lyu M R, King I. A classification-based approach to question routing in community question answering[C]// Proceedings of the 21st international conference companion on World Wide Web. ACM, 2012:783-790.

[31] Hu H, Zhang H, Xuan J, et al. Effective Bug Triage Based on Historical Bug-Fix Information[C]// IEEE, International Symposium on Software Reliability Engineering. IEEE, 2014:122-132.

[32] Choetkiertikul M, Avery D, Dam H K, et al. Who Will Answer My Question on Stack Overflow?[C]// Software Engineering Conference. IEEE, 2015:155-164.

[33] Zhao W X, Jiang J, Weng J, et al. Comparing Twitter and Traditional Media Using Topic Models[M]// Advances in Information Retrieval. Springer Berlin Heidelberg, 2011:338-349.

[34] Titov I, Mcdonald R. Modeling online reviews with multi-grain topic models[J]. 2012:111-120.