Only Recommend to You: Towards Personalized Models for Question Recommendation in Community Question Answering*

Xin LIAN, Xiangyu HU, Haiwei ZHANG, Xinyu CHEN, Xiaojie YUAN*

Institute of Computer Science and Technology, Nankai University, Tianjin 300071, China

Abstract

Question Answering communities such as Yahoo! Answers have emerged as a popular medium for online information seeking and knowledge sharing. However, as these QA sites always have thousands of new questions posted daily, it’s time-consuming for users to find the questions that are of interest to them. Some QA sites conduct question recommendation by category filtering. Category filtering is efficient but not very effective. While smart use of non-textual features is crucial in many web services, there has been little research to develop systematic and formal approaches to process these features. To solve the problem, we present a machine learning approach to predict whether a user will be interested in an unsolved question after category filtering. Whether a question is recommended to a user depends on the user’s prior experience, expectations, and personal preferences. We develop personalized models, formalize the problem, and explore a variety of content, structure, and interaction features for this task using standard machine learning techniques. The experimental results show that our approach leads to a better performance than other baseline approaches and increases the F-measure by a factor ranging from 15 to 20%..

Keywords: Community Question Answering; Question Recommendation; Personalized Model; Machine Learning

1 Introduction

Community Question Answering (CQA) has become a popular medium for online information seeking and knowledge sharing. In contrast to search engines, questions in the CQA are posted in the form of natural language and attract more specific users to answer. In the last few years, many CQA systems have been launched, including Yahoo! Answers\(^1\), BuyAns\(^2\), Live QnA\(^3\). Since their inception, CQA sites have rapidly gained popularity. Hundreds of millions of answers have

---

\(^1\)http://answers.yahoo.com
\(^2\)http://buyans.com
\(^3\)http://qna.live.com

*Project supported by the National Nature Science Foundation of China (No. 61170184).
*Corresponding author.
Email address: forwarding82@gmail.com (Xiaojie YUAN).
already been posted for tens of millions of questions in Yahoo! Answers. Unfortunately, it’s
time-consuming for users to find the questions that are of interest to them. Users often focus on
two or three fields, but they have to decide which field to browse this time. As a result, large
quantities of questions are in the state of no response and askers would have to wait for a long
time before getting answers. Question recommendation is to help users find interesting questions
and expedite the answering of new questions.

Many previous approaches can be classified into the three categories. 1) Content-based rec-
ommendation [1, 2]: The methods assume that the user has high interest in new question if
he/she has answered many similar questions before. Their target is to find similar questions.
The methods are based on similarity algorithms, which have a lot of theory basic and classical
models. Nevertheless, the methods focus on text contents, weakening the structure information of
forums. 2) Collaborative Recommendation[6, 7]: The methods assume that the similar users may
interest in the same questions which are often used to recommend in the field of movie, music,
books and so on. They cluster people into groups with similar interest. Computing similarities
between people and movies/music/books allows to recommendation or not. However, users may
interest in two or three fields, the complexity of clustering and low precision are big challenges.
3) Authority-based recommendation[8, 9]: The methods’ target to find domain experts by user
networks. Experts indeed improve the quality of the answers. The disadvantage is the number
of experts is limited. They are not able to deal with tens of millions of questions. In addition,
they ignore the content of the posts and rank users globally. The low participation rate of users
in CQA service is the crucial problem which limits its development potential. In a word, most
researches are based on text statistics, not making full use of the features of CQA. Especially, the
precision is low when the users’ history data is little.

To tackle this problem, we present a machine learning approach after category filtering. The
category information of questions is proven to be positive information for question answering
recommendation[5, 10, 11]. Sina iAsk and Yahoo! Answers recommend the unsolved questions to
the answerer based on category analysis. This paper makes use of category filtering to reduce the
number of candidates for question recommendation. Some researchers have solved the problems
in the CQA by adapting machine learning techniques, including user satisfaction[12, 13],questions
popularity[14], predicting the best answer[15]. They got higher precision and less dependency on
the data size. We explore structure features of a forum and give a detailed analysis. Then we
choose an appropriate classification algorithm. The experimental results show our approach leads
to a better performance than other baseline approaches and increases the F-measure by a factor
ranging from 15 to 20%.

The rest of this paper is organized as follows. Section 2 reviews some prior work related to our
approach. Section 3 details the proposed framework including models, features and classifiers.
Section 4 reports on the experimental evaluation. At last, we conclude the paper and discuss
about the future work in Section 5.

2 Related Work

Content-based recommendation can be divided into question search and expert search. Question
search is to return questions semantically equivalent or close to a given question. Duan et al.[2]
proposed to conduct question search by identifying question topic and question focus. Expert
search is to estimate the probability p(u|q) of a user being an expert for a given question based
on the previous question answering of the user. M. Qu et al.[1] adopted the Probabilistic Latent Semantic Analysis (PLSA) model for question recommendation. PLSA model is known for its ability of capturing underlying topics.

Authority-based recommendation is to discover users’ authorities by user networks, which is also called expert finding. Jurczyk et al.[8] and Zhang et al.[9] evaluated link algorithms PageRank and HITS to rank users based on their authority scores. The difference is that Zhang et al. is applied to a small data set. Some researches[4, 3] integrated the contents of posts and users networks to rank users.

Some researchers have solved some problems in the CQA by adopting machine learning techniques, which is closely related to our work. Liu et al.[12, 13] proposed towards personalized models for predicting satisfaction in CQA. Features were organized around the basic entities in a question answering community: questions, answers, question-answer pairs, users and categories. Closest to our work, Sun et al.[14] designed a ‘majority-based perception algorithm’ and explored only two aspects of features: features about question and features about asker to predict question popularity. Our work differs from it on predicting question popularity for a user, not for all users. We explored more features and gave rational analysis.

3 Question Recommendation

In this section, we present our approach to address the problem of question recommendation by the means of machine learning. It consists of four steps: (a) propose a new scheme, which is illustrated in Figure 1; (b) give problem definition and basic assumption; (c) learn features; (d) explore some families of classification algorithms.

3.1 Scheme

Users can play four different roles in a community question answering: asker, answerer, voter and searcher. The paper focuses on the roles of asker and answerer. Askers post questions. Answerers answer questions. A user can be both an asker and an answerer. However, the
community rules forbid askers from answering their own questions, and then a user would be either an asker or an answer for a question. For conciseness, we use roles to describe users in a question and answer.

The scheme of our approach is shown in Figure 1. The sold line shows the system behavior, the dotted line shows users’ behavior. It consists of three main components: Information Storage, Category Filtering and Personalized Classification Model. Information Storage stores questions, answers and the operation history of each user, including posting a question, answering a question, refusing a recommendation. For a recommended user, the Category Filtering component selects candidate questions according to the category of the questions that the user has just answered. The Personalized Classification Model component chooses a machine learning training algorithm to build a personalized model with the features of the user’s history information. The model assigns each candidate question to recommendation category or not and adjusts with the feedback from the user.

3.2 Problem statement

The problem is described as a two-class classification problem: recommendation or no-recommendation, without the distinction of the possibly recommendation. For a pair of user $u$ and question $q$, the problem is defined as:

$$c(u, q) \rightarrow \text{recommendation/\text{non-recommendation}}$$

The best way of evaluating recommendation is the evaluation of the recommended users. However, there's no evaluation information in the initial state of question recommendation system and it's hard to find those answerers to cooperate our research. Therefore, we give a basic assumption. We assume that users are interested in the questions that they answered while they are uninterested in the last and next questions which near the questions that they answered. The assumption is based on an observation. Questions are often displayed by the way of list in a category. There are about 20 questions on a page of questions list. Users scan two or three questions once, and then choose interesting questions to answer. We don’t confirm the number of questions that users scan once. But the questions, nearest to the questions that are answered by the users, must be seen. If users don’t choose to answer, it shows that they are uninterested or unable to answer. Such questions shouldn’t be recommended to the users.

**Definition 1** A user in a CQA is considered to be interested in a question if the user scans and answers the question. Otherwise, the user is considered to be uninterested if the user doesn’t answer after scanning it. The question which nearest to the question answered belongs to the uninterested situation if it isn’t answered by the user.

3.3 Learning features

There are four basic entities in a community question answering. They are questions, answers, askers, answerers. The paper is towards personalized. In a personalized model, the answerer features are all the same. Therefore, the portion of answerers can be omitted. We will use Yahoo! Answers as the example to describe our approach although our approach can be applied to the questions from other CQA sites as well. We derived a set of 18 different features for
questions, answers and askers entities. These features are listed as follows, each followed by a brief explanation of its underlying rationale. They have obvious personalized characteristics.

- **Category**: CQA sites build corresponding categories and subcategories according to users’ interest fields, although the number and the name of categories are not the same between the CQA sites. The behavior of a user in different categories may be different as well.

- **QuestionLength**: Answerers see questions title first when they scan questions list. Statistics show that most answerers prefer to longer question title.

- **QuestionType**: QuestionType indicates the information type that askers need, which can be divided into who, where, when, what, which, why, how and yes/no.

- **DescriptionLength**: The lengths of question descriptions are very different. Statistics show that some answerers prefer to no description or elaborate description (hundreds of words), while some answerers prefer to simple description (tens of words).

- **QuestionStars**: In Yahoo! Answers, if users regard a question as interesting, they can add a star to the question no matter whether they answer it or not. The number of stars reflects the popularity of a question.

- **AnswersNumber**: AnswersNumber is the number of answers before an answerer answers. The reason why we choose the number of answers before an answerer answers is to simulate the situation when the answers receive the question.

- **VotesNumber**: If users support or oppose to an existed answer, they can give a positive vote (thumbs up) or a negative vote (thumbs down). VotesNumber is the number of votes on the answers before an answerer answers. It reflects the popularity of a question and the quality of the existed answers.

- **AskerProfilePhoto**: Yahoo! Answers supplies some system profile photo. Statistics show that users who use a personalized profile photo post or answer questions actively.

- **AskerBestRate**: The best rate is the rate of the user’s answers being regarded as the best answer for users’ answers. It indicates askers’ authority. In General, the more specific askers attract the more answerers.

- **AskPoint and AskerPointRate**: In Yahoo! Answers, users get points by answering, commenting, voting and so on except posting questions. AskPoint is the total points from registration. AskerPointRate is the average points every day. Therefore, more points indicate more active in answering.

- **AskerQuestionsNumber and AskerAnswersNumber**: Askers who post/answer more questions are more experienced. They may give more accurate question description, more attractive question title.

- **AskerQuestionsStars**: AskerQuestionsStars is the total number of question stars for an asker’s all posted questions.

- **Textual Features**: We derive word n-gram (unigram and bigram) features from the text of the question. We ignore the stop words and useless words in the unigram features. Textual feature selection is based on Information Gain (IG).
3.4 Classification algorithms

We conducted a set of experiments with some families of classification algorithms: Decision trees, Support Vector Machines, Boosting, Naive Bayes and Logistic regression, all using the implementations in the Weka\cite{16} framework.

- **Decision Trees**: A decision tree is a decision support tool that uses a tree-like graph or model of decisions. Decision trees model is built according to the training data set. We apply the Weka implementation of J48.

- **SVM**: An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are predicted to belong to a category based on which side of the gap they fall on. We apply the Weka implementation of SMO.

- **Naive Bayes**: A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes’ theorem with strong (naive) independence assumptions. It assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. We apply the Weka implementation of NaiveBayes.

- **Boosting-based**: Boosting is a machine learning meta-algorithm for performing supervised learning. Most boosting algorithms consist of iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier. When they are added, they are typically weighted in some way that is usually related to the weak learners’ accuracy. We apply the Weka implementation of AdaBoost.

- **Logistic regression**: In statistics, logistic regression (sometimes called the logistic model or logit model) is a type of regression analysis used for predicting the outcome of a binary dependent variable (e.g. "yes" vs. "no") based on one or more predictor variables. We apply the Weka implementation of SimpleLogistic. It incorporates attribute selection by fitting simple regression functions in LogitBoost.

4 Experimental Evaluation

We now describe the measures used for the evaluation, the dataset and the experimental results. We want to recommend current question to a user who is interested in the question. In other words, we predict whether the user will be interested for a given user and current question.

4.1 Dataset and evaluation metrics

Our data was based on a snapshot of Yahoo! Answers, crawled in the early 2012. There are 26 top-level categories at Yahoo! Answers, such as “Arts & Humanities”, “Education & Reference”. We randomly selected 100 users whose activities are public in each top-level category, 2600 users in total. Then we collected the questions that have been answered by the 100 users in the latest two months and last/next questions that are assumed to be uninterested in by the users in the definition 1. Statistics on the data sets is shown in Table 1. Even though ours is formally a two-class classification problem, we primarily focus on the recommended or positive class. The reason for this is that we have higher certainty about the true positive likelihood of our recommended labels compared to the non-recommended more properly to be stated as unknown cases.
We made use of three measures for evaluating the experimental results. They are Precision, Recall and F1. Precision is the fraction of the predicted interesting questions that were indeed answered by the user. Recall is the fraction of all answered questions that were correctly recommended by the system. F1 is the geometric mean of Precision and Recall measures, computed as $\frac{2PR}{P+R}$.

4.2 Methods compared

We now describe the baseline, the representations of our specific methods and compared methods in other papers. The reason why we didn’t choose other methods is that they need manual annotation. The paper [4] proposed three methods. The first one is based on users’ profile, the second one is based on latent topic, the last one is based on the clustering of users. The former two methods got higher performance. Therefore, we choose the two methods to compare and parameters are set according to [4]. It adopted probability to rank the recommended users and recommended a question to the top-k users. The Precision of top-k users is the percentage of the top-k candidate answers retrieved that are correct. In our dataset, the number of users who can be recommended to a question is at most 5. Therefore, the Recall of top-k users is the same as the definition in the section 4.1. The experiment shows that the performance in the situation of top-5 outperforms that of top-10. We compare with the two methods in the situation of top-5.

- **QRM_C4.5**: Our system implementing a decision tree using the C4.5 algorithm.
- **QRM_SVM**: Our system implementation using the SVM classifier.
- **QRM_Boosting**: Our system implementing the AdaBoost algorithm.
- **QRM_NB**: Our system implementing the Naive Bayes classifier.
- **QRM_SL**: Our system implementing the SimpleLogitstic classifier.
- **Profile**: A system computing the user expertise with Profile-based model.
- **Thread**: A system computing the user expertise with Thread-based model.

4.3 Experimental results

First, we report the main classification results of the paper. Second, we choose the best one to compare with Profile, Thread, varying with the number of users’ history data. Finally, to gain a better understanding of the important features for this domain, we report the top 10 non-textual features and textual features with highest Information Gain. To derive text features, we use the Lucene[17] to preprocess the question text, including tokenization, stop words filtering and stemming. Table 2 shows the recommendation accuracy for the different implementations of QRM, in particular comparing the choice in classifier algorithm and feature sets (whether to use the textual features). In comparison, QRM_SL results in the best performance of all the classification variants, while QRM_NB results in the worst performance. QRM_NB
assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. Its low performance shows that dependencies between features exist. QRM_SLogistic builds a logistic regression model using LogitBoost. LogitBoost minimizes the logistic loss and places less emphasis on examples that are very badly classified. LogitBoost is more appropriate when there is noise in the labels. Our features are numeric, which SimpleLogistic is very fit for. Hence, in the next experiment, we choose QRM_SLogistic to compare with other methods. Figure 2 reports the Precision, Recall and F1 for Profile, Thread, QRM_SL and QRM_SL+Text with varying number of previous questions answered. The abscissa shows the number of questions answered for training. Then we use users’ latest 10 questions (5 answered and 5 unanswered) to test. QRM_SL and QRM_SL+Text methods outperform Profile and Thread contrast methods especially with less than 30 previous questions answered. For QRM_SL+Text, textual information become helpful for users with more than 30 previous questions answered. The contrast methods have sensible trend of ascent with the number of questions answered. It’s because those methods are based on probability statistics which depends on the quantities of data. Nevertheless, 30 questions in training are sufficient to achieve F1 of 0.61 for our methods. Therefore, our methods are more effective with small quantities of data and increase the F-measure by a factor ranging from 15 to 20%.

5 Conclusions

This paper describes a personalized model to conduct question recommendation in the CQA. We formalized the problem, explored a variety of content, structure, and interaction features for this task using standard machine learning techniques and gave a brief explanation of these features’ underlying rationale. Experimental results on real data from Yahoo! Answers show that the proposed approaches can effectively recommend questions. Especially, our methods outperform methods based on probability statistics with small quantities of data. We increase the F-measure by a factor ranging from 15 to 20%.

Table 2: Comparison of QRM with different classifiers

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Without Text</th>
<th>With Text</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>QRM_C4.5</td>
<td>0.65</td>
<td>0.66</td>
</tr>
<tr>
<td>QRM_NB</td>
<td>0.64</td>
<td>0.59</td>
</tr>
<tr>
<td>QRM_SVM</td>
<td>0.62</td>
<td>0.72</td>
</tr>
<tr>
<td>QRM_SLogistic</td>
<td>0.65</td>
<td>0.73</td>
</tr>
<tr>
<td>QRM_Boosting</td>
<td>0.65</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Fig. 2: The scheme of question recommendation
References


