Question Quality Analysis and Prediction in Community Question Answering Services with Coupled Mutual Reinforcement

Jinwei Liu, *Student Member, IEEE*, Haiying Shen, *Senior Member, IEEE*, and Lei Yu

Abstract—Community question answering services (CQAS) (e.g., Yahoo! Answers) provides a platform where people post questions and answer questions posed by others. Previous works analyzed the answer quality (AQ) based on answer-related features, but neglect the question-related features on AQ. Previous work analyzed how asker- and question-related features affect the question quality (QQ) regarding the amount of attention from users, the number of answers and the question solving latency, but neglect the correlation between QQ and AQ (measured by the rating of the best answer), which is critical to quality of service (QoS). We handle this problem from two aspects. First, we additionally use QQ in measuring AQ, and analyze the correlation between a comprehensive list of features (including answer-related features) and QQ. Second, we propose the first method that estimates the probability for a given question to obtain high AQ. Our analysis on the Yahoo! Answers trace confirmed that the list of our identified features exert influence on AQ, which determines QQ. For the correlation analysis, the previous classification algorithms cannot consider the mutual interactions between multiple (>2) classes of features. We then propose a novel Coupled Semi-Supervised Mutual Reinforcement-based Label Propagation (CSMRLP) algorithm for this purpose. Our extensive experiments show that CSMRLP outperforms the Mutual Reinforcement-based Label Propagation (MRLP) and five other traditional classification algorithms in the accuracy of AQ classification, and the effectiveness of our proposed method in AQ prediction. Finally, we provide suggestions on how to create a question that will receive high AQ, which can be exploited to improve the QoS of CQAS.

Index Terms—Community question answering, question quality, prediction, classification, answer quality

1 Introduction

Search engines are widely used in our daily life to find the answers to questions. However, they require users to know the effective search keywords to questions, without which users may spend an extremely long time in searching for answers [35]. Since some user questions are typically personal, heterogeneous, extremely specific and open-ended, search engines are usually not intelligent enough to find a single web page that can directly answer such questions [36]. Since real humans are believed to understand and answer better than a machine [41], community question answering services (CQAS) provide a platform to allow people to post questions and answer questions posed by others. In a CQAS, a question is open for receiving answers during a certain time period. An asker can select the best answer for his/her question along with a rating in a given range (e.g., [1], [5]). Also, each question has an attribute of "tag-of-interests", which represents the number of users interested in this question.

Although CQAS provides more effective ways to find answers by using human resources, it is found that a large portion of questions remain unanswered in such systems, referred to as question starvation phenomenon [23], [36]. To solve this problem, it is important to improve the question quality (QQ) in order to attract more attention to a question from users. A previous work [22] measured QQ with consideration of the amount of attention from users, the number of answers and the question solving latency. However, this work neglects the correlation between QQ and AQ, which is critical to QoS. Agichtein et al. [3] showed that QQ correlates with AQ. In addition to the number of responses and response latency, AQ also is a critical factor that determines the quality of service (QoS) of CQAS. Thus, we argue that QQ measurement should also consider AQ. As a result, enhancing QQ will automatically improve AQ and hence QoS.

In this work, we consider the correlation between QQ and AQ from two aspects. First, we additionally use QQ in measuring AQ, and analyze the correlation between a comprehensive list of features (including answer-related features) and QQ. Second, we propose the first method that estimates the probability for a given question to obtain high AQ. Our work has more advantages over the work in [22] also in that we identified more features in the asker- and question-related types that affect QQ. Though many previous works analyzed AQ, they focus on the analysis of the influence from the answer-related features (e.g., answer length, reference inclusion), which cannot be retrieved until the answers are posted. While our work can proactively predict AQ upon the posted question, it also provides suggestions on how to create a question that can receive a high AQ.

Our analysis of the Yahoo! Answers trace confirmed that the list of our identified features exert influence on AQ,

• J. Liu and H. Shen are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634.
E-mail: {jinweil, shenh}@clemson.edu.
• L. Yu is with the Department of Computer Science, Georgia State University, Atlanta, GA 30302. E-mail: lyu13@student.gsu.edu.

*Manuscript received 21 Feb. 2015; revised 22 May 2015; accepted 8 June 2015. Date of publication 18 June 2015; date of current version 7 Apr. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSC.2015.2446991*

which determines QQ. For the correlation analysis, as shown in Fig. 1(a) [22], the previous classification algorithms cannot consider the mutual interactions between multiple (>2) classes of features. We propose a novel Coupled Semi-Supervised Mutual Reinforcement-based Label Propagation (CSMRLP) algorithm for this purpose. Our extensive experiments show that CSMRLP outperforms MRLP and five other classic classification algorithms in the accuracy of AQ classification, and the effectiveness of our proposed method in AQ prediction. Finally, we provide suggestions on how to create a question to receive high AQ, which can be exploited to improve the QoS of CQAS.

In order to identify the factors influencing QQ, we conducted statistical analysis on a trace from Yahoo! Answers [1], and extract four kinds of features: question-, asker-, category- and answer-related features (see Fig. 1(b) [22]). For the correlation analysis, it is important to consider the mutual interactions between multiple classes of features. However, traditional classification algorithms cannot consider such mutual interactions, while MRLP [22] cannot consider the mutual interactions between multiple (>2) classes of features. To handle this problem, we propose a graph-based Coupled Semi-Supervised Mutual Reinforcement algorithm that considers the four kinds of features to accurately predict QQ, i.e., to classify it into either the high quality class or the low quality class. CSMRLP views the interactions in CQAS as composite bipartite graphs and exploits mutual reinforcement between the connected entities in each bipartite graph. We conducted extensive trace-driven experiments to compare the performance of CSMRLP with MRLP [22] as well as five other traditional classification algorithms. The experiment results show that CSMRLP outperforms other methods in terms of different metrics, which also indicates our four features together can better represent QQ.

The contribution of the paper is summarized as follows:

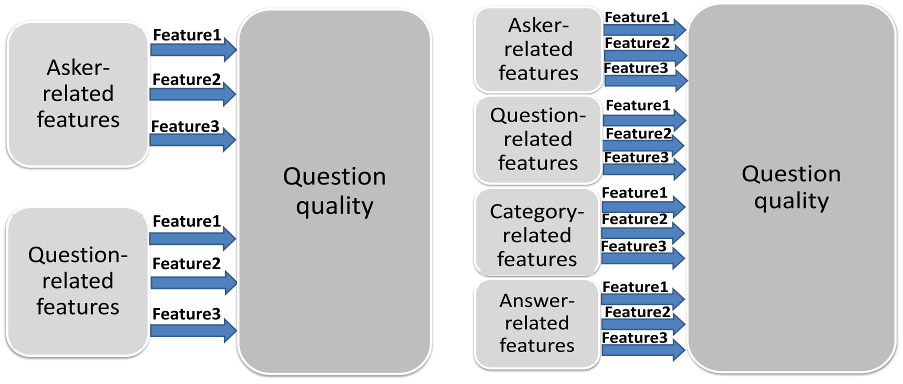
- We re-defined QQ and analyzed the correlation between various factors and QQ using our crawled data from Yahoo! Answers to identify the factors affecting QQ. Based on our analysis results, we identified a comprehensive list of four kinds of features: question-, asker-, category- and answer-related features that are closely related to QQ for QQ prediction.
- Based on the features highly related to QQ, we propose a novel graph-based Coupled Semi-Supervised Mutual Reinforcement (CSSL) algorithm CSMRLP to predict QQ.
- We evaluate CSMRLP's classification performance for QQ and compare it with MRLP and other various classic classification algorithms. The experiment results demonstrate the advantages of CSMRLP.
- By modeling the interaction between answerers and answers, the interaction between questions and answers, and the interaction between categories and questions and answers, we propose an algorithm Coupled Semi-Supervised Mutual Reinforcement-based Label Propagation Answer Quality (CSMRLPAQ) for answer quality (AQ) estimation.
- In the view of the features of a question, we provide guides on how to ask high quality questions that have a higher probability of receiving high quality answers.

The remainder of this paper is organized as follows. Section 2 presents a review of the related work. Section 3 introduces preliminaries and objective. Section 4presents the ground truth and our analysis on the crawled data from Yahoo! Answers. Section 5 describes the prediction methods. Section 6 presents the experimental results. Section 7 concludes our work with remarks on our future work.

2 Related Work

Question answering roles in the context of the community Q&A site have been examined. Harper et al. [14] investigated predictors of answer quality through a comparative, controlled field study of responses provided across several online Q&A sites. Recently, much research has been conducted on CQAS in order to enhance the QoS of CQAS. Some works focus on finding similar questions in the archive for a given question to retrieve historical high quality answers for the question. Jeon et al. [16] presented a translation-based retrieval model to find similar questions in a large question and answer archive. Wang et al. [38] analyzed the syntactic tree matching method and used it to find similar questions. Shtok et al. [36] analyzed question starvation, and proposed to find the similar resolved questions to reduce the number of unanswered questions. Identifying experts in communities to answer questions helps increase AQ. Bouguessa et al. [7] proposed a model based on InDegree which is the number of best answers provided by users to identify experts. Zhang et al. [42] analyzed data from an online forum, seeking to discover users with high expertise. Pal et al. [28] explored approaches to identify potential experts as early as within the first two weeks of their association with the CQAS. Riahi et al. [31] built a profile for each expert and used the profiles to find experts in community question answering. Yang et al. [40] proposed Topic Expertise Model (TEM) and CQARank to find the right experts, retrieve archived similar questions and recommend best answers to new questions. Some works aim to identify one or more answers from a list of candidate answers that semantically answer the corresponding question [39]. Ko et al. [19] proposed a unified probabilistic answer ranking model to simultaneously address the answer relevance and answer similarity problems. Wang et al. [39] proposed a method to rank community answers by modeling question-answer relationships via analogical reasoning. Asker satisfaction plays crucial role in the growth or decay of a CQA. Liu et al. [25] presented a general prediction model to predict asker satisfaction from the perspective of information seeker, which is in contrast to the more traditional relevance-based assessment.

Little previous research payed attention to question [18], [34], which directly affects AQ [3]. Shah et al. [34] investigated question quality among questions posted in Yahoo! Answers to assess what factors contribute to the goodness of a question and determine if poor quality questions could be flagged. Kitzie et al. [18] explored the relationship between determinants of question quality as provided by human assessors (non-textual features), and features mechanically extracted from the question content (textual features), and found that textual features make a significant contribution to explaining assessments of question quality. Li et al. [22] proposed a Mutual Reinforcement-based Label Propagation



(a) Two types of features in previous work [22] (b) Four types of features in our work [22]

Fig. 1. Comparison of features designed in previous work and our work.

(MRLP) to predict QQ. Their model only considers asker- and question-related related features. Also, they neglect AQ in measuring QQ [6]. Unlike these previous works, however, our work is innovative because i) it is the first that can predict AQ upon question posting; ii) we identified a comprehensive list of factors that affects AQ by comprehensive trace data analysis; and iii) we designed the first classification algorithm that can consider the interactions between multiple (>2) types of features in classification.

Classification algorithms are important for quality prediction based on different features. Mutual Reinforcement-based Label Propagation proposed in [22] can only consider the interaction between two types of features to predict QQ in CQA service. There are many traditional classification algorithms. *Naïve Bayes* [11], [21] is a simple, fast, yet surprisingly effective method. It outperformed many sophisticated classifiers over a large number of datasets, especially where the features are not strongly correlated. *Decision Trees* [5], [27] can handle different types of features and they have been used successfully in many real-word situations. We choose two implementations of decision tree [30], C4.5 algorithm and RandomForest, to perform classification, and compare our proposed algorithm with these two algorithms. J48, the Weka implementation of the famous C4.5 algorithm, is used (Quinlan 1996) [5]. *Boosting* has been shown to be quite effective for many tasks (e.g., Freund and Schapire 1996) [33]. Specifically, we use the Weka implementation of AdaBoost AdaBoost.M1 (AdaBoostM1 in short) [13] to effectively classify QQ and AQ. Support vector machine (SVM) [10], [26], [37] constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which is used for classification, regression, or other tasks. Support vector machines are widely used classifier because of its robustness in the presence of noise, and high reported accuracy. In this paper, we use the Weka implementation of SMO [25], [29] to efficiently handle large training sets without requiring a quadratic programming solver. However, the traditional classification algorithms cannot consider the interaction between different types. Our CSMRLP classification algorithm is novel because it can consider the interactions between ≥ 2 types for classification.

3 Definition and Objective

In this section, we first introduce the preliminaries, our definitions and assumptions, and then introduce our objective of this work. In Yahoo! Answers, an asker needs to pay five points for asking one question, and an answerer receives two points for answering a question, and receives 10 points if his/her answer is chosen as the best answer. The system assigns stars to each user based on his/her answering activity, and also gives each user a level ([1], [7]) based on the user activity, duration of membership, etc.

Definition 1 (Asker satisfaction). *An asker in a QA community is satisfied with the service if and only if the following conditions are satisfied: 1) (s)he personally closed the question, 2) (s)he chose the best answer, 3) (s)he provided a ≥ 3 rating for the best answer. This rating is referred to as* asker rating *for short.*

The rating score of the best answer reflects the asker's degree of satisfaction.

Definition 2 (Question Quality). *Question Quality is defined as "well-formedness, readability, utility and interestingness" [3], andrefers to its ability to attract users' attention [22], gain answering attempts, and improve the response timeliness and quality of the best answer.*

We set the range of QQ to [0, 1]. To convert the rating to a two-class decision, we consider the questions in the range [0.6, 1] as high quality questions and those in the range [0, 0.6) as low quality questions. We regard answers with a rating ≥ 3 as good answers and those with a rating ≤ 2 as poor answers.

Definition 3 (Answer Quality). *Answer Quality refers to the ability that a given answer will be chosen as the best answer indicating the responsiveness, accuracy, and comprehensiveness of the answer to a question [6]. The answer is considered to be a high-quality answer, if it 1) is selected as the best answer, and 2) receives a rating ≥ 3.*

Our problem is to evaluate QQ, and then estimate the probability that the question will obtain a high-quality answer. The final goal of our work is to answer the question how to generate a question (in the view of the question features) to obtain a high-quality answer.

4 Data Description and Analysis

In this section, we first describe our dataset, then we provide the description of ground truth for question quality, providing the baseline for the following studies and

TABLE 1
Summary of Questions with High Quality and Low Quality Levels

Quality Level	High Quality	Low Quality
Count	6,138	3,862

TABLE 2
Statistics of the Sample Data Crawled from Yahoo! Answers

Questions	Answers	Askers	Categories
10,000	57,931	3,922	26

analyses. Finally, we analyze the trace data and observe that category-, asker-, and answer-related features influence QQ.

4.1 Dataset

In order to get representative data for our experimental analysis from Yahoo! Answers, we crawled all the questions posted between May and June 2013, and randomly sampled 10,000 questions from 26 categories posted during that time period. Table 2 shows the statistics of the sample data. We also crawled the profile page of each user and saved the statistics of his questions and answers to study the historical features of users. Basically, a user's profile contains the following information: user level, the number of stars received, the number of points earned, the number of questions asked, the number of questions resolved, the number of questions answered, and the number of those answers that were chosen as the best answers. Each user has a "Member since" attribute, which means the time period since (s)he registered for the CQAS as a user.

4.2 Ground Truth

We set the ground truth using human assessments. Specifically, we used Amazon's paid rater service, Mechanical Turk (MTurk) [2], [17], to assess question quality by asking MTurk workers to rate questions. We used a 3-scale rating method for all rating tasks, {good=1, medium=2, bad=3}. Each question is labeled by five-seven workers. MTurk workers were provided the following guidelines:

10,000 questions will be given to you. Each question has been posted to the social question and answering site Yahoo! Answers. Yahoo! Answers is a CQAS, and anyone can sign up and participate by asking questions, answering questions, voting on the best answer to a question, giving comments to questions, and other activities.

We would appreciateif you would identify a question asgood, medium, or bad. A question is considered as good if you think it can be answered and it can clearly communicate the asker's information need; a question is considered as medium if you think it can be answered but it cannot clearly indicate the asker's information need; a question is considered as bad if you think it cannot be answered and it cannot clearly indicate the asker's information need.

In order to validate the labels of MTurk workers, we asked 10 researchers to label the quality for all questions. Then we analyzed the agreement between the MTurk workers and the researchers. We first calculated the average rating by researchers and the average rating by MTurk workers for each question, respectively, then we used a threshold S_{th} to cast each average numerical rating score dr into a binary rating br (if $dr \leq S_{th}$ then $br =$ high quality, otherwise $br =$ low quality), and finally we computed the Fleiss's kappa coefficient [12] based on these binary ratings between the two sources. The highest kappa value 0.47 was achieved with a threshold of 1.5 (average agreement is around 0.75). The analysis shows that the ratings from MTurk workers are reasonable.

From the above agreement analysis, we find that although the kappa coefficient among MTurk workers is not very high (possibly due to the careless rating of some MTurk workers), the average rating by MTurk workers shows a moderate agreement with researchers. Therefore, we use the average rating by MTurk workers as our ground truth. Finally, we acquired the distribution of question quality based on the MTurk and 10 researchers. Table 1 summarizes questions with high quality and low quality levels.

4.3 Data Processing

To make the data more amenable for modeling, we used a two-stage preprocessing, performed on each feature:

1. We first parsed the raw text of each question, and obtained the numerical values of features by counting or summation. For example, we counted the words in a question's subject to get the numerical value of the feature "subject length". For binary features, such as "question's references inclusion", we used binary variable to represent the numerical value of the feature.
2. For each feature, we normalized the numerical value of the feature to a range between 0 and 1.

4.4 Data Analysis

4.4.1 Relationship between the QQ and Categories

Table 3 illustrates the detailed statistics of the top 10 most frequent categories that are from Yahoo! Answers [1], [9] in our dataset. We can find that the questions in these categories comprise almost 77 percent of all questions in the dataset. Question ratio of a category is the percent of the questions in this category among the questions in all 26 categories. In particular, the *Entertainment & Music category* is the most popular category containing 16.22 percent of all the questions and drawing 7.5 answers per question. *Family & Relationships* also gains high popularity, which contains 12.77 percent of all the questions and draws 5.1 answers per question. *Politics & Government* contains 6.49 percent of all the questions and draws 8.61 answers per question. *Science & Mathematics* contains only 4.59 percent of all the questions and draws 7.76 answers per question. Also, though asker rating is usually high as it is the rating of the best answer, it still varies between the categories. *Entertainment & Music* and *Sports* gain the highest asker rating, while *Family & Relationships* the lowest. We see that the number of answers per question for the three categories are 7.50, 3.91 and 5.10, respectively, which means that this metric and asker rating may not have corresponding relationships. In short, AQ , one of the determinants of QQ [6], and other statistics of the questions vary among different categories, so category also affects QQ, and it should be considered when evaluating QQ [14].

TABLE 3
Detailed Statistics for the Top 10 Most Frequent Categories in Our Dataset

Category	Questions	Answers	Answers per question	Question ratio	Asker rating
Entertainment & Music	1,622	12,170	7.50	16.22%	4.56
Family & Relationships	1,277	6,508	5.10	12.77%	4.30
Society & Culture	1,114	8,922	8.01	11.14%	4.41
Health	789	3,159	4.00	7.89%	4.42
Politics & Government	649	5,589	8.61	6.49%	4.51
Pregnancy & Parenting	493	2,600	5.27	4.93%	4.39
Science & Mathematics	459	3,562	7.76	4.59%	4.45
Sports	435	1,701	3.91	4.35%	4.56
Computers & Internet	408	2,213	5.42	4.08%	4.38
Pets	397	1,355	3.41	3.97%	4.44
Cumulative (10 Categories)	7,643	47,779	6.25	76.43%	4.44
Overall (26 Categories)	10,000	57,931	5.79	100%	4.34

4.4.2 Relationship between the QQ and Askers

We use the average rating by MTurk workers to represent each question's quality, then we calculate the average QQ based on questions' average ratings by MTurk workers. Finally, we normalize the average QQ to a range between 0 and 1. We analyze the relationship between an asker's QQ and his features. Fig. 2a shows the correlation between the average QQ of users and the number of user points. The users are ranked based on the number of their points on the X axis. We see that the average QQ increases as the user points increase. The more points an asker has earned, the more likely the asker will ask high-quality questions. Fig. 2b shows the correlation between the average QQ of each user and his level. The users are clustered and ranked based on their levels on the X-axis. We see that the average QQ increases as the user level increases. The higher the level of an asker, the higher the quality of questions he asks. Fig. 2c shows the correlation between the average QQ and user "Member since". The users are ranked based on their "Member since" on the X-axis. We find that the average QQ of users increases as the user "Member since" increases. The longer an asker stays in the system, the higher the probability that he will ask high-quality questions, which may possibly due to more QA activity experiences.

4.4.3 Relationship between the QQ and AQ

Fig. 3 shows the average QQ versus AQ. We use the average rating by MTurk workers to represent QQ, and we normalize QQ to a range between 0 and 1. The AQ is asker's rating of the best answer of the question. From the figure, we can find that the average QQ increases as AQ increases, and vice versa. It confirms our conjecture: low-quality questions are likely to receive poor answers while high-quality questions are more likely to attract good answers. QQ determines AQ to a certain extent.

To further investigate the relationship between QQ and AQ, we first calculate the Pearson correlation between QQ and AQ for the users of top K levels, then we compute the Pearson correlation between QQ and AQ for the top K users ranked by users' points. Fig. 4a shows the Pearson correlation between QQ and AQ for the users of top K levels. In Fig. 4a, we can find that QQ is related to AQ. Fig. 4b reports the Pearson correlation between QQ and AQ for the top K users ranked by users' points. From Fig. 4b, we can also find that QQ is related to AQ.

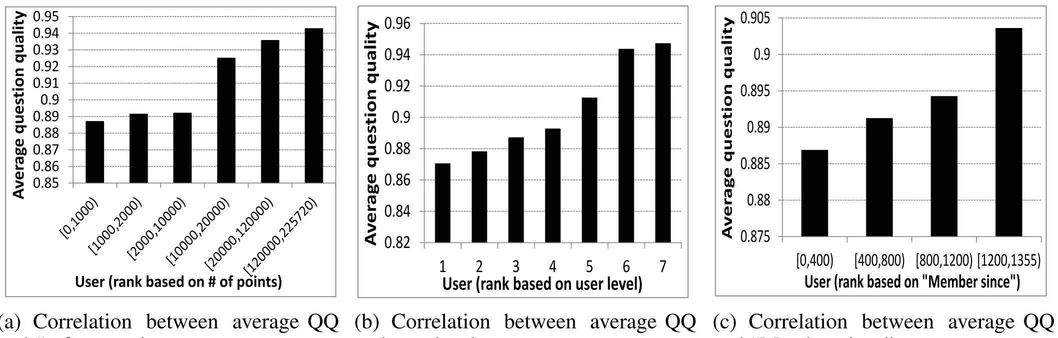
5 Prediction Method

In the above section, we observed that category-, asker-, and answer-related features influence QQ. Then, in this section, we propose a method that considers these three types of features and question-related features to predict the QQ. We first identify the features in each type (Section 5.1.1), and then propose a method to use these identified features in QQ prediction (Section 5.2).

5.1 Features

5.1.1 Question

We first looked to previous works on CQAS (such as [22], [44]) to find as many possible features of each type. Table 4



(a) Correlation between average QQ and # of user points (b) Correlation between average QQ and user level (c) Correlation between average QQ and "Member since"

Fig. 2. Relationship between AQ and asker's profile.

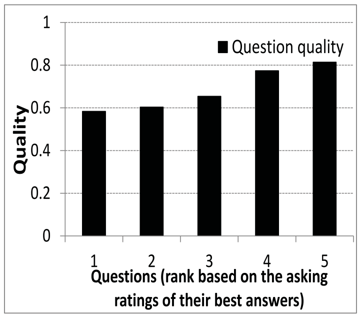
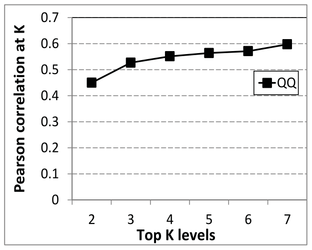
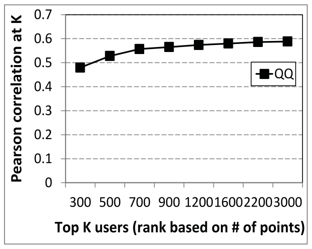


Fig. 3. Correlation between QQ and AQ.



(a) Pearson correlation between QQ and AQ for top K levels



(b) Pearson correlation between QQ and AQ for top K users

Fig. 4. Pearson correlation at K for QQ versus AQ for all categories.

illustrates the features we found and their detailed description. Compared to previous work [22], our work not only develops two more kinds of features that affect QQ, but also designs some new features belonging to question-related features and asker-related features. For example, prior asker-answerer interaction (the social interaction between askers and answerers) is an important social component of Q&A sites (especially social Q&A sites). Long-time askers and answerers (who have a long member since) are likely to have interacted many times before, and some Q&A systems allows friending/following/favoring particular users so others can receive notifications of new content they post. Thus, users who have social interactions are likely to answer each other's questions. Therefore, we added the feature of

TABLE 4
A List of Features in Each Type of Question-, Asker-, Category-, and Answer-Related Features

Feature	Description
	Question-related features
Subject length	Number of words in the question's subject
Content length	Number of words in the question's content
Number of answers [6]	Number of answers the question received
Number of comments	Number of comments given by other participants
Number of polite words	Number of polite words in question's content
Category matching	How well the question matches its current category in CQAS
Capital error	Number of capital errors
Post time	Time (in hours) of the day when the question was posted
Wh-type	Wh-word introducing the question title (e.g.,"what","where", etc.)
Total positive/negative votes	Total number of positive/negative votes for the answers
Average of positive/negative votes	Average number of positive/negative votes for the answers
Max positive/negative votes	Max number of positive/negative votes for the answers
Most_vote answer positive/negative votes	Number of positive/negative votes for the answer which received most votes
QA overlap	Words shared between question and answer which received most votes
Additional details	Question's additional details
Question's references inclusion	Including references or not in the question
	Asker-related features
Asker's points	The total points earned from the beginning of the registration time
Asker's level	A skill level indicating the asker's level of skill in the area
Question resolved	Number of question resolved in the past for the asker
Total answers received	Number of all answers the asker received in the past
Best answers received	Number of best answers the asker received in the past
Question asked	Number of questions the asker asked in the past
Points earned recently	Number of points the asker earned in the recent week
Asker's stars	Number of stars the asker received
Member since [24]	How long since the asker's last registration
Best answer ratio	Ratio of the asker's best answer over all answers the asker posted
Question/Answer ratio	Ratio of questions to answers the asker posted in the past
Prior asker-answerer interaction	Number of prior interactions between askers and answerers
	Category-related features
Average answers per question	Average number of answers per question for that category
Average asker rating	Average rating given by asker for an answer from that category
Average voter rating	Average votes given by voters for an answer from that category
Average number of questions per hour	Average number of questions per hour from that category
	Answer-related features
Rep. rank of best answer [4]	Reciprocal rank of the best answer in the list of answers for a question
Best answer length [15]	The length of the best answer's content
Answer's reference inclusion	Including references or not in the answer

TABLE 5
Notations

Notation	Description
$\mathcal{U}$	The set of askers
$\mathcal{Q}$	The set of questions
$\mathcal{C}$	The set of categories
$\mathcal{A}$	The set of answers
U	The vector of askers' asking expertise
Q	The vector of question quality
A	The vector of answer quality
C	The vector of categories
E	A matrix mapping users and their questions
$w_q(q_i, q_j)$	The weight for the edge linking question q_i and question q_j
N	The probabilistic transition matrix, where N_{ij} is the probability of transit from q_i to q_j
$w_u(u_i, u_j)$	The weight for the edge linking user u_i and user u_j
M	A probabilistic transition matrix, where M_{ij} is the probability of transit from u_i to u_j
I	A matrix mapping questions and the categories to which they belong
λ_q	The weighting parameter for questions
λ_u	The weighting parameter for users

TABLE 6
Top 20 Features with Highest Information Gain

IG	Feature
0.233010	Q: Additional details
0.224303	Q: Category matching
0.208208	A: Rep. rank of best answer
0.207116	U: Asker's stars
0.198888	U: Best answer ratio
0.193148	U: Best answers received
0.190095	U: Asker's points
0.184800	C: Average asker rating
0.000601	C: Average voter rating
0.000432	A: Best answer length
0.000416	U: Asker's level
0.000232	C: Average answers per question
0.000231	Q: Question's references inclusion
0.000230	A: Answer's references inclusion
0.000203	Q: Question's number of comments give by other participants
0.000195	Q: Subject length
0.000194	Q: The length of question's content
0.000155	Q: Number of answers
0.000143	Q: Wh-type
0.000129	Q: Number of capital errors

prior asker-answerer interaction for question quality evaluation. To get the value of the feature, we used *asker/answerer overlap* to measure the feature. As in the work [32], we use $\mathcal{N}^-(u)$ to denote the set of *in-neighbors* of user u and use $\mathcal{N}^+(u)$ to denote the set of *out-neighbors* of user u, where the in-neighbor of u are users who answered questions from u, and the out-neighbors are users whose questions are answered by u. Then, we use $|\mathcal{N}^-(u) \cap \mathcal{N}^+(u)|$ to calculate the value of asker/answerer overlap.

To identify the features that influence the QQ, and determine the degree of influence of each feature on the QQ, we computed the information gain (IG) of each feature in Table 4. Then, we ranked the features based on their IGs and listed the top 20 features in Table 6. Based on this data, we see that the most important feature for creating a high quality question is "Additional details" which is not considered in [22]. "Additional details" captures the accuracy, comprehensiveness and completeness of the information provided by the question. Questions with these additional details can help users understand the problem better, hence provide better quality answers. "Category matching" is also very important for generating high quality question because good category matching clearly communicates the question's topic, and the most relevant potential answerers can easily see the question. "Rep. rank of best answer" is also important in determining QQ, and it indicates how easily a question can achieve the best answer. User related features (e.g., "Asker's stars", "Best answer ratio", "Best answers received", "Asker's points", "Asker's level") play an important role in determining QQ. This is consistent with our previous observations and confirms our previous conjecture. In addition, category- and answer-related features (e.g., "Average asker rating", "Average voter rating", "Best answer length", "Average answers per question", "Answer's references inclusion") also play a more important role in determining QQ.

Given the IGs mentioned above, there are also other practical things that an asker should do in writing a high quality question. "Question's references inclusion" is important because a question with reference(s) included can help users better understand the background of the problem described in the question, and users have more chance to provide a good quality answer based on their better understanding of the problem. "Wh-type" feature is also important because "Wh-type" uses concise text and helps users understand questions quickly. "Subject length" and "Question's content length" are also important. The subject or the content of a question should not be too short to fully communicate the information, but it should also not be so long (i.e., >500 words) that potential answerers will not read it. Fewer capitalization errors and proper grammar are also important. A low number of capitalization errors and proper grammar help the users with context. After we identified the features that have higher influences on QQ (e.g., higher than a IG threshold), we use these features to predict the QQ. The details of the prediction are presented in Section 5.2.

5.2 Prediction Model

We propose a graph-based CSSL algorithm called CSMRLP to predict QQ in CQAS. This algorithm abstracts all influences from the identified features in each asker-, question-, category- and answer-related type to *asker expertise*, *QQ*, *category reputation* and *AQ*. Then, it uses bipartite graph to mathematically formulate the problem of learning these four abstracted categories, and finally calculate the QQ.

Let $\mathcal{U}$ be the set of askers, $\mathcal{Q}$ be the set of questions, $\mathcal{C}$ be the set of categories, and $\mathcal{A}$ be the set of answers. The identified features for QQ prediction form the feature space of questions ($\mathcal{X}(\mathcal{Q})$), of answers ($\mathcal{X}(\mathcal{A})$), of askers ($\mathcal{X}(\mathcal{U})$) and of categories ($\mathcal{X}(\mathcal{C})$). Table 5 provides the description for each notation used in CSMRLP. We construct the sets of bipartite graphs that share the same question to be coupled. We use a bipartite graph $G_{ij} = \{N_{ij}, E_{ij}\}$ to model the relation between asker expertise and QQ. N_{ij} is the set of nodes and E_{ij} is the set of undirected edges. N_{ij} consists of two types of nodes that correspond to the questions and the askers, respectively. There is an edge between a question node and an asker node if the question is asked by the asker. Below, we explain how the model works. We use a bipartite graph to represent the relationship between questions (quality) and askers (askers' asking expertise). Suppose there are m askers who ask n questions. Let U denote the vector $(m \times 1)$ of askers' asking expertise, and $Q(n \times 1)$ denote the vector of QQ. Define a $m \times n$ matrix E, where $e_{ij} = 1(i \in [1, m], j \in [1, n])$ means u_i asks q_j, otherwise $e_{ij} = 0$. Then we can get E' from E

$$E'_{ij} = \frac{e_{ij}}{\sum_{k=1}^{n} e_{ik}}. \tag{1}$$

For the question part of the bipartite graph, an edge connects any two questions (i.e., q_i, q_j) if they are in the same category. The weight for the edge linking q_i and q_j is represented by $w_q(q_i, q_j)$, which is calculated based on the cosine similarity between the features of two questions x_i and x_j:

$$w_q(q_i, q_j) = exp\left(-\frac{\|x_i^q - x_j^q\|^2}{\lambda_q^2}\right), \tag{2}$$

where λ_q is a weighting parameter, $w_q(q_i, q_j)$ is set to be 0 if q_i and q_j belong to two different categories. In addition, $w_q(q_i, q_i) = 0$.

Define an $n \times n$ probabilistic transition matrix N:

$$N_{ij} = P(q_i \to q_j) = \frac{w_q(q_i, q_j)}{\sum_{k=1}^{n} w_q(q_i, q_k)}, \tag{3}$$

where N_{ij} is the probability of transit from q_i to q_j. An edge connects any two askers (i.e., u_i, u_j) who have asked questions in the same category for asker part of the graph. The weight for the edge linking u_i and u_j is represented by $w_u(u_i, u_j)$, which is calculated based on the cosine similarity between the features of two askers (asker-related features) x_i^u and x_j^u:

$$w_u(u_i, u_j) = exp\left(-\frac{\|x_i^u - x_j^u\|^2}{\lambda_u^2}\right), \tag{4}$$

where λ_u is a weighting parameter, $w_u(u_i, u_j)$ is set to be 0 if u_i and u_j have not asked questions belonging to the same category. In addition, $w_u(u_i, u_i) = 0$.

Then, we define an $m \times m$ probabilistic transition matrix

$$M_{ij} = P(u_i \to u_j) = \frac{w_u(u_i, u_j)}{\sum_{k=1}^{m} w_u(u_i, u_k)}. \tag{5}$$

Given some known (labeled) examples of U and Q. The following equation can be used to estimate the askers' asking expertise from their neighbors and their questions' qualities

$$U_{c+1} = \alpha M U_c + (1 - \alpha) E' Q_c. \tag{6}$$

Correspondingly, the equation below can be used to estimate questions' quality from their neighbors and askers' asking expertise

$$Q_{c+1} = \beta N Q_c + (1 - \beta) E^T U_{c+1}. \tag{7}$$

Repeating k (the number of iterations) times, all the questions' quality and askers' asking expertise can be estimated.

Algorithm 1 shows the pseudocode for iteratively finding QQ and asker's asking expertise. The algorithm first propagates user expertise by estimating askers' asking expertise from their neighbors and their questions' qualities (line 5). Second, the algorithm propagates QQ by estimating questions' quality from their neighbors and askers' asking expertise (line 6). Then it clamps the labeled data of askers' asking expertise and questions' qualities (line 7). After repeating a certain number of times, the algorithm can estimate all questions' qualities and askers' asking expertise.

Algorithm 1. Pseudocode for Iteratively Finding QQ and Asker's Asking Expertise

```
1 Input: user asking expertise vector $U_0$, QQ vector $Q_0$, $E$, transition matrixes $M$ and $N$, weighting coefficients $\alpha$, $\beta$, some manual labels of $U_0$ and/or $Q_0$.
2 Output: Questions' qualities $Q$ and askers' asking expertise $U$.
3 Set c = 0.
4 while not convergence do
5     Propagate user expertise. $U_{c+1} \leftarrow \alpha M U_c + (1 - \alpha) E' Q_c$.
      // Formula (6)
6     Propagate QQ. $Q_{c+1} \leftarrow \beta N Q_c + (1 - \beta) E^T U_{c+1}$.
      // Formula (7)
7     Clamp the labeled data of $U_{c+1}$ and $Q_{c+1}$.
8     Set $c = c + 1$.
9   return $Q, U$
```

Suppose there are s categories. Denote $C(s \times 1)$ as the vector of categories. Then we define an $s \times n$ matrix H, where $h_{ij} = 1(i \in [1, s], j \in [1, n])$ means question q_j is in category C_i, otherwise $h_{ij} = 0$. We can get H' from H

$$H'_{ij} = \frac{h_{ij}}{\sum_{k=1}^{n} h_{ik}}. \tag{8}$$

Similarly, we can get

$$C_{c+1} = \gamma O C_c + (1 - \gamma) H' Q_c, \tag{9}$$

and

$$Q_{c+1} = \eta P Q_c + (1 - \eta) H^T C_{c+1}, \tag{10}$$

where O and P are transition matrixes. We can get QQ after a certain number of iterations.

Algorithm 2. Pseudocode for Iteratively Finding QQ and the Reputation of Categories

```
1 Input: category-related features vector $C_0$, QQ vector $Q_0$, $H$, transition matrixes $O$ and $P$, weighting coefficients $\gamma$, $\eta$, some manual labels of $C_0$ and/or $Q_0$.
2 Output: Questions' qualities $Q$ and reputation of categories $C$.
3 Set c = 0.
4 while not convergence do
5     Propagate category reputation.
      $C_{c+1} \leftarrow \gamma O C_c + (1 - \gamma) H' Q_c$. // Formula (9)
6     Propagate QQ. $Q_{c+1} \leftarrow \eta P Q_c + (1 - \eta) H^T C_{c+1}$.
      // Formula (10)
7     Clamp the labeled data of $C_{c+1}$ and $Q_{c+1}$.
8     Set $c = c + 1$.
9 return $Q, C$
```

Algorithm 2 shows the pseudocode for iteratively finding QQ and category reputation. The algorithm first propagates category reputation by estimating the reputation of categories from their neighbors and their questions' qualities (line 5). Second, the algorithm propagates QQ by estimating questions' qualities from their neighbors and the reputation of categories (line 6). Then it clamps the labeled data of reputation of categories and questions' qualities (line 7). After repeating a certain number of times, the algorithm can estimate all questions' qualities and reputation of categories.

Suppose there are u answers. Denote $A(u \times 1)$ as the vector of AQ. Then, we define a $u \times n$ matrix I, where $i_{ij} = 1 (i \in [1, u], j \in [1, n])$ means answer a_i is the answer of question q_j, otherwise $i_{ij} = 0$. We can get I' from I

$$I'_{ij} = \frac{i_{ij}}{\sum_{k=1}^{n} i_{ik}}. \tag{11}$$

Similarly, we can get

$$A_{c+1} = \mu R A_c + (1-\mu) I' Q_c, \tag{12}$$

and

$$Q_{c+1} = \lambda T Q_c + (1-\lambda) I^T A_{c+1}, \tag{13}$$

where R and T are transition matrixes. We can get the QQ after a certain number of iterations.

Algorithm 3 shows the pseudocode for iteratively finding QQ and AQ. The algorithm first propagates AQ by estimating answers' qualities from their neighbors and their questions' qualities (line 5). Second, the algorithm propagates QQ by estimating questions' quality from their neighbors and answers' qualities (line 6). Then it clamps the labeled data of answers' qualities and questions' qualities (line 7). After repeating a certain number of times, the algorithm can estimate all questions' qualities and answers' qualities.

Algorithm 3. Pseudocode for Iteratively Finding QQ and AQ

```
1 Input: AQ vector $A_0$, QQ vector $Q_0$, $I$, transition matrixes $R$
  and $T$, weighting coefficients $\mu$, $\lambda$, some manual labels of $A_0$
  and/or $Q_0$.
2 Output: Questions' qualities $Q$ and answers' qualities $A$.
3 Set c = 0.
4 while not convergence do
5     Propagate AQ. $A_{c+1} \leftarrow \mu R A_c + (1-\mu) I' Q_c$.
      // Formula (12)
6     Propagate QQ. $Q_{c+1} \leftarrow \lambda T Q_c + (1-\lambda) I^T A_{c+1}$.
      // Formula (13)
7     Clamp the labeled data of $A_{c+1}$ and $Q_{c+1}$.
8     Set $c = c + 1$.
9 return $Q$, $A$
```

Denote Q_1^* as the QQ estimated based on asker-related or question-related features. Denote Q_2^* as the QQ estimated based on category-related or question-related features. Denote Q_3^* as the QQ estimated based on answer-related or question-related features. Thus we have

$$Q = \Sigma_{i=1}^{m} w_i Q_i^*, \tag{14}$$

where $m = 3$, and w_i $(i = 1, \ldots, m)$ are the weights.

To determine the weight of each factor (w_i) we use relative comparison method. Let $H = (h_{ij})_{m \times m}, i, j = 1, 2, \ldots, m$

$$h_{ij} = \begin{cases} 1, & if\ Q_i^*\ is\ more\ important\ than\ Q_j^* \\ 0.5, & if\ Q_i^*\ is\ the\ same\ importance\ as\ Q_j^* \\ 0, & if\ Q_i^*\ is\ less\ important\ than\ Q_j^*. \end{cases} \tag{15}$$

Obviously:

$$\begin{cases} h_{ii} = 0.5 \\ h_{ij} + h_{ji} = 1. \end{cases} \tag{16}$$

Since the weight of each factor is related to the practical application, largely depending on the goal of the system, we can qualitatively assign weights to Q_1^*, Q_2^*, Q_3^* according to the goal of the system and determine the order of weight for each factor. The weight can be calculated by the following formula:

$$w_i = \frac{\sum_{j=1}^{m} h_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{m} h_{ij}}. \tag{17}$$

In this way, we quantify the qualitative weight of each factor in Formula (14). Based on the result from the work [22], we prove the convergence of CSMRLP in below.

Theorem 5.1. *CSMRLP is convergent as the number of iterations goes to infinity.*

Proof. Suppose there are a labeled data and b unlabeled data for the qualities of questions in c_k (the kth category), and r labeled data and s unlabeled data for askers' asking expertise in c_k, i.e., $Q^{c_k} = [q_1^{c_k}, \ldots, q_a^{c_k}, q_{a+1}^{c_k}, \ldots, q_{a+b}^{c_k}]^T$ and $U^{c_k} = [u_1^{c_k}, \ldots, u_r^{c_k}, u_{r+1}^{c_k}, \ldots, u_{r+s}^{c_k}]^T$. For analytical tractability, we split the matrices E', E^T, M and N into four parts based on labeled data and unlabeled data. □

For category c_k, given some known labels of U^{c_k} and/or Q^{c_k}, Formulas (6) and (7) can be written as

$$\begin{bmatrix} U_r^{c_k} \\ U_s^{c_k} \end{bmatrix}_{c+1} = \alpha \begin{bmatrix} M_{rr} & M_{rs} \\ M_{sr} & M_{ss} \end{bmatrix} \begin{bmatrix} U_r^{c_k} \\ U_s^{c_k} \end{bmatrix}_c + (1-\alpha) \begin{bmatrix} E'_{ra} & E'_{rb} \\ E'_{sa} & E'_{sb} \end{bmatrix} \begin{bmatrix} Q_a^{c_k} \\ Q_b^{c_k} \end{bmatrix}_c \tag{18}$$

and

$$\begin{bmatrix} Q_a^{c_k} \\ Q_b^{c_k} \end{bmatrix}_{c+1} = \beta \begin{bmatrix} N_{aa} & N_{ab} \\ N_{ba} & N_{bb} \end{bmatrix} \begin{bmatrix} Q_a^{c_k} \\ Q_b^{c_k} \end{bmatrix}_c + (1-\beta) \begin{bmatrix} E_{ra}^T & E_{sa}^T \\ E_{rb}^T & E_{sb}^T \end{bmatrix} \begin{bmatrix} Q_r^{c_k} \\ Q_s^{c_k} \end{bmatrix}_c, \tag{19}$$

respectively. We only need to consider $U_s^{c_k}$ and $Q_b^{c_k}$ because $U_r^{c_k}$ and $Q_a^{c_k}$ are labeled data and they are clamped to manual labels in each iteration. Based on Formulas (18) and (19), we have

$$\begin{bmatrix} U_s^{c_k} \\ Q_b^{c_k} \end{bmatrix}_{c+1} = \alpha \begin{bmatrix} \alpha M_{ss} & (1-\alpha) E'_{sb} \\ (1-\beta) E_{sb}^T & \beta N_{bb} \end{bmatrix} \begin{bmatrix} U_s^{c_k} \\ Q_b^{c_k} \end{bmatrix}_c + \begin{bmatrix} \alpha M_{sr} U_s^{c_k} + (1-\alpha) E'_{sa} Q_a^{c_k} \\ \beta N_{ba} Q_a^{c_k} + (1-\beta) E_{rb}^T U_r^{c_k} \end{bmatrix}. \tag{20}$$

Let $A = \begin{bmatrix} \alpha M_{ss} & (1-\alpha)E'_{sb} \\ (1-\beta)E_{sb}^T & \beta N_{bb} \end{bmatrix}$,

$d = \begin{bmatrix} \alpha M_{sr} U_r^{c_k} + (1-\alpha)E'_{sa} Q_a^{c_k} \\ \beta N_{ba} Q_a^{c_k} + (1-\beta)E_{rb}^T U_r^{c_k} \end{bmatrix}$, we have

$$\begin{bmatrix} U_s^{c_k} \\ Q_b^{c_k} \end{bmatrix}_n = A^n \begin{bmatrix} U_s^{c_k} \\ Q_b^{c_k} \end{bmatrix}_0 + (\textstyle\sum_{i=1}^n A^{i-1})d, \tag{21}$$

where $[\begin{smallmatrix} U_s^{c_k} \\ Q_b^{c_k} \end{smallmatrix}]_0$ are the initial values for unlabeled askers' asking expertise and questions' qualities, and n indicates the number of iterations. Based on the works [22], [43], $\exists$ $\rho < 1$ such that $\sum_{j=1}^{s+b} A_{ij} \leq \rho$ $(\forall i \in \{1, 2, \ldots, s+b\})$, and therefore $A^n[U_s^{c_k}, Q_b^{c_k}]_0^T \to 0$. Eventually, we can get the following fixed values:

$$\begin{bmatrix} U_s^{c_k} \\ Q_b^{c_k} \end{bmatrix} = (1-A)^{-1}d. \tag{22}$$

Hence, the convergence of Algorithm 1 is proved.

Similarly, we suppose there are a labeled data and b unlabeled data for the qualities of questions, and r labeled data and s unlabeled data for category reputations, we also split the matrices H', H^T, O and P into four parts based on labeled data and unlabeled data

Let $B = \begin{bmatrix} \gamma O_{ss} & (1-\gamma)H'_{sb} \\ (1-\eta)H_{sb}^T & \eta P_{bb} \end{bmatrix}$,

$e = \begin{bmatrix} \gamma O_{sr} C_r + (1-\gamma)H'_{sa} Q_a \\ \eta P_{ba} Q_a + (1-\eta)H_{rb}^T C_r \end{bmatrix}$.

Based on the works [22], [43], $\exists$ $\varepsilon < 1$ such that $\sum_{j=1}^{s+b} B_{ij} \leq \varepsilon$ $(\forall i \in \{1, 2, \ldots, s+b\})$, and therefore $B^n[C_s, Q_b]_0^T \to 0$. Eventually, we can get the following fixed values:

$$\begin{bmatrix} C_s \\ Q_b \end{bmatrix} = (1-B)^{-1}e. \tag{23}$$

Hence, the convergence of Algorithm 2 is proved.

Similar to the approach of proving Algorithm 1, we assume there are a labeled data and b unlabeled data for the qualities of questions in category c_k, and r labeled data and s unlabeled data for answer qualities in c_k

Let $D = \begin{bmatrix} \mu R_{ss} & (1-\mu)I'_{sb} \\ (1-\lambda)I_{sb}^T & \lambda T_{bb} \end{bmatrix}$,

$f = \begin{bmatrix} \mu R_{sr} A_r^{c_k} + (1-\mu)I'_{sa} Q_a^{c_k} \\ \lambda T_{ba} Q_a^{c_k} + (1-\lambda)I_{rb}^T A_r^{c_k} \end{bmatrix}$.

Based on the works [22], [43], $\exists$ $\delta < 1$ such that $\sum_{j=1}^{s+b} D_{ij} \leq \delta$ $(\forall i \in \{1, 2, \ldots, s+b\})$, and therefore $D^n[A_s^{c_k}, Q_b^{c_k}]_0^T \to 0$. Eventually, we can get the following fixed values:

$$\begin{bmatrix} A_s^{c_k} \\ Q_b^{c_k} \end{bmatrix} = (1-D)^{-1}f. \tag{24}$$

Hence, the convergence of Algorithm 5.2 is proved.

Let $\sigma = max\{\rho, \varepsilon, \delta\}$. Thus $\exists$ $\sigma < 1$, $\sum_{j=1}^{s+b} A_{ij} \leq \sigma$, $\sum_{j=1}^{s+b} B_{ij} \leq \sigma$, $\sum_{j=1}^{s+b} D_{ij} \leq \sigma$, $(\forall i \in \{1, 2, \ldots, s+b\})$. Hence, $A^n[U_s^{c_k}, Q_b^{c_k}]_0^T \to 0$, $B^n[C_s, Q_b]_0^T \to 0$, $D^n[A_s^{c_k}, Q_b^{c_k}]_0^T \to 0$. Thus, the convergences of Algorithm 5.2, Algorithm 5.2 and Algorithm 5.2 can be achieved simultaneously.

Therefore, the convergence of CSMRLP is proved.

6 Performance Evaluation

In this section, we first describe the metrics used for evaluation, then present how to setup the experiments, and finally present our experimental results.

6.1 Evaluation Metrics

Our classification is formally a two-class classification problem. We primarily focus on the high quality (or positive) class. The reason behind this is that we have higher certainty about the true positive likelihood of our high-quality labels compared to the low-quality labels. For completeness, we measure Accuracy and Specificity. Specifically, we measure the Precision, Recall, and F1 for the high-quality class, and measure Specificity and the overall Accuracy for the low-quality class and both classes, respectively.

- *Precision:* the fraction of the predicted high-quality questions that indeed received a high-quality answer (rated as high-quality by the asker), computed as $\frac{TP}{TP+FP}$, where TP represents True Positive, and FP represents False Positive.
- *Recall:* the fraction of all rated high-quality questions that were correctly identified by the system, computed as $\frac{TP}{TP+FN}$, where FN represents False Negative.
- *F1:* the geometric mean of Precision (P) and Recall (R) measures, computed as $\frac{2PR}{P+R}$.
- *Accuracy:* the overall fraction of instances classified correctly into the proper class, computed as $\frac{TP+TN}{TP+TN+FP+FN}$, where TN represents True Negative. Often, Accuracy is not the right metric when the class distribution is skewed.
- *Specificity:* the fraction of all rated low quality questions that were correctly identified by the system, computed as $\frac{TN}{FP+TN}$.

In the experiments, we will mainly focus on predicting the high-quality class, thus we will rely more on Precision, Recall and F1 rather than the overall Accuracy or Specificity [25].

6.2 Experiment Setup

We crawled questions posted between May and June 2013 from Yahoo! Answers, and randomly sampled 10,000 questions from 26 categories. The dataset for evaluating our proposed method is the randomly sampled 10,000 questions. We used those 20 features with the highest information gain to evaluate the performance of the baseline methods and our proposed method. We separated the dataset into two parts: training data and testing data. Training rate means the percentage of the dataset used as training data. For example, a 90 percent training rate means that 90 percent of the dataset is used for training and 10 percent of the dataset is used for testing. Then, we performed classification using CSMRLP, MRLP and five traditional algorithms including

TABLE 7
Parameter Settings

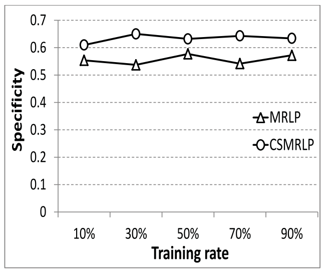
Parameter	Meaning	Setting
$\|\mathcal{U}\|$	# of askers	7,685
$\|\mathcal{Q}\|$	# of questions	10,000
$\|\mathcal{C}\|$	# of categories	26
$\|\mathcal{A}\|$	# of answers	57,931
α	A certain coefficient $\in (0,1)$	0.5
β	A certain coefficient $\in (0,1)$	0.5
γ	A certain coefficient $\in (0,1)$	0.5
η	A certain coefficient $\in (0,1)$	0.5
μ	A certain coefficient $\in (0,1)$	0.5
λ	A certain coefficient $\in (0,1)$	0.5
ω_1	Weight for QQ estimated based on asker- or question-related features	0.5
ω_2	Weight for QQ estimated based on category- or question-related features	0.25
ω_3	Weight for QQ estimated based on answer- or question-related features	0.25

one of the best performing classifiers SVM in literature, respectively. We used the Weka implementation of SMO [25], [29] to represent SVM in performance evaluation. To avoid overfitting, we performed classification 10 times at each training rate for each algorithm, then we used the mean value of each evaluation metric as the algorithm's performance on this metric. In addition, we varied the training rate, and we repeatedly divided the dataset into training and testing sets using cross-validation with percentage split and performed classification multiple times at each training rate. Table 7 shows the parameter settings in our experiment unless otherwise specified. For the weights w_1, w_2 and w_3, we used Formulas (15)-(17) to set the values based on the number of question-, asker-, category-, and answer-related features included in top 20 features.

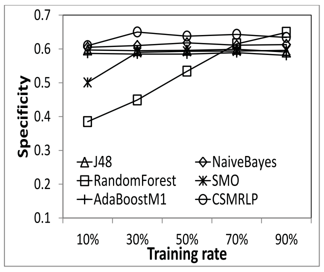
6.3 Experimental Results

Fig. 5a shows the precision rate versus the training rate in CSMRLP and MRLP. The Precision of CSMRLP does not change much as the training rate increases, and it constantly stays at around 0.847. The Precision of MRLP varies from about 0.354 to 0.459. It first increases as the training rate increases, and reaches its highest value with a training rate of 50 percent. Then it begins to decrease slowly as the training rate increases to 90 percent. Fig. 5b shows the recall rate versus the training rate in CSMRLP and MRLP. The Recall of CSMRLP is consistent at around 0.857 and its variance is very small. The recall rate of MRLP varies from about 0.333 to 0.456. It changes as the training rate increases. Fig. 5c shows the F1 versus training rate in CSMRLP and MRLP. The F1 of CSMRLP does not change much as the training rate increases, and it is stabilized at about 0.848. The F1 of MRLP varies from about 0.343 to 0.457. It first increases as the training rate increases, and it reaches its highest value with a training rate of 50 percent. Then it starts decreasing slowly as the training rate increases to 90 percent. From Figs. 5a, 5b, and 5c, we find that CSMRLP has higher values of Precision, Recall and F1 than MRLP. This indicates CSMRLP outperforms MRLP in these metrics. Fig. 5d shows the Accuracy versus training rate in CSMRLP and MRLP. The Accuracy of CSMRLP is stabilized about 0.842 and the changes are small. The Accuracy of MRLP varies from about 0.594 to 0.637. As the training rate increases, Accuracy first decreases slightly. After the training rate increases to 50 percent, Accuracy then begins to increase. Finally it reaches its highest value with a training rate of 90 percent. From Fig. 5d, we can find that the Accuracy of CSMRLP is always higher than that of MRLP. This, again, suggests CSMRLP outperforms MRLP on Accuracy.

Fig. 6a shows the Specificity versus the training rate in CSMRLP and MRLP. The Specificity of CSMRLP stays nearly constant at 0.635 and its variance is small. It first slightly increases as the training rate increases, reaching its highest value at a training rate of 30 percent, and then remains nearly constant. Compared with CSMRLP, the Specificity of MRLP is lower than that of CSMRLP, and it has a relatively larger fluctuation. Since the class distribution is skewed and the positive class accounts for the majority, CSMRLP generates Specificity with smaller fluctuation, which indicates CSMRLP performs better than MRLP. Fig. 6b compares the Specificity of CSMRLP with other classic classification algorithms. We find that the Specificity of CSMRLP is higher than that of others, and it has a small

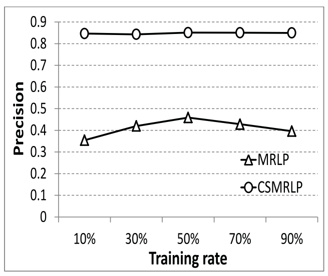


(a) Specificity in CSMRLP and MRLP

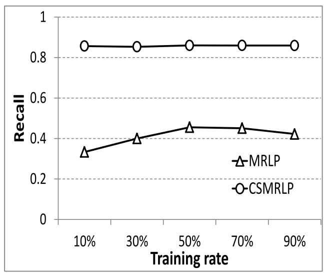


(b) Specificity in CSMRLP and five other classic classification algorithms

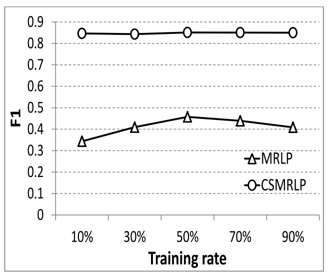
Fig. 6. Specificity versus training rate.



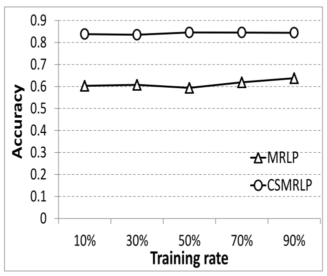
(a) Precision



(b) Recall



(c) F1



(d) Accuracy

Fig. 5. Performance of various evaluation metrics versus training rate in CSMRLP and MRLP.

TABLE 8
Summary of Various Evaluation Metrics for Question Quality versus Training Rate across Different Methods

	Training rate					Training rate				
	10%	30%	50%	70%	90%	10%	30%	50%	70%	90%
	Precision					Recall				
NaiveBayes	0.753	0.768	0.774	0.771	0.778	0.839	0.845	0.845	0.849	0.845
J48	0.728	0.731	0.730	0.724	0.737	0.853	0.855	0.855	0.851	0.859
RandomForest	0.803	0.851	0.886	0.893	0.914	0.838	0.859	0.887	0.893	0.917
SMO	0.759	0.758	0.749	0.776	0.735	0.846	0.852	0.851	0.850	0.858
AdaBoostM1	0.735	0.734	0.733	0.731	0.740	0.853	0.855	0.855	0.851	0.859
CSMRLP	0.843	0.851	0.850	0.850	0.848	0.856	0.854	0.860	0.859	0.859
	F1					Accuracy				
NaiveBayes	0.784	0.791	0.793	0.787	0.798	0.742	0.749	0.750	0.750	0.753
J48	0.785	0.788	0.788	0.782	0.793	0.755	0.758	0.759	0.762	0.758
RandomForest	0.800	0.830	0.859	0.870	0.905	0.742	0.759	0.773	0.783	0.807
SMO	0.787	0.788	0.788	0.784	0.793	0.746	0.754	0.759	0.758	0.761
AdaBoostM1	0.788	0.788	0.789	0.785	0.793	0.753	0.755	0.759	0.755	0.759
CSMRLP	0.846	0.843	0.851	0.850	0.850	0.759	0.763	0.771	0.773	0.773

fluctuation. The Specificity of RandomForest significantly increases as the training rate increases, whereas the Specificities of other algorithms do not change much. NaiveBayes generates the second highest Specificity in all of the algorithms. CSMRLP produces a higher Specificity (with a small fluctuation) than all of the other algorithms even when the class distribution is skewed and the positive class takes the largest proportion, which suggests that CSMRLP performs better than the other algorithms.

Table 8 compares the performance of various evaluation metrics of CSMRLP with those of other classic classification algorithms. Table 8 shows the Precision (top left) versus training rate using CSMRLP, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. From Table 8, we can find the Precision of CSMRLP is higher than that of RandomForest when the training rate equals or is slightly larger than 10 percent. After the training rate increases to 50 percent, the Precision of RandomForest is slightly higher than CSMRLP. Both CSMRLP and RandomForest have a Precision greater than 0.84. RandomForest, as a supervised method, highly depends on training rate [8], [20], and it achieves higher Precision only if the training rate is large enough. This limits the applicability of RandomForest to new sites and domains. However, CSMRLP is not dependent on training rate, and it can achieve high Precision at lower training rate. This is because CSMRLP is a semi-supervised coupled mutual reinforcement framework, which can simultaneously calculate content quality (QQ and AQ), asking expertise, category reputation and thus requires relatively few labeled examples [6]. The Precisions of J48, NaiveBayes, SMO and AdaBoostM1 are lower than that of CSMRLP. This suggests that CSMRLP can achieve higher Precision without depending on the training rate. This demonstrates that CSMRLP performs better than most of the classic classification algorithms. More importantly, it achieves higher Precision without relying on training rate, which overcomes the limitation of the supervised approaches. Table 8 shows the Recall (top right) versus training rate in CSMRLP, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. In Table 8, it is obvious that RandomForest has the highest Recall when the training rate is larger than 30 percent. It is highly dependent on training rate, and it increases as the training rate increases. CSMRLP has the second highest Recall, and it is not dependent on the training rate. The recall values of the other algorithms are roughly the same value. This indicates CSMRLP has a good performance on Recall, and it outperforms most of the other classic algorithms. Table 8 shows the F1 (bottom left) versus training rate in CSMRLP, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. From Table 8, we find the F1 of CSMRLP is almost the highest among all of the algorithms when the training rate is less than 50 percent. After the training rate increases to 50 percent, the F1 value of CSMRLP is slightly lower than RandomForest, but it is still much higher than those of the other algorithms. The F1 of CSMRLP is about 0.848 at different training rates, but the F1 of RandomForest is much lower than CSMRLP when the training rate is less than or equal to 30 percent. RandomForest achieves higher F1 only if the training rate is large enough and thus it is highly dependent on training rate, but CSMRLP is not dependent on training rate, and it can therefore achieve high F1 at lower training rate due to the same reason explained for precision. The F1 values of J48, NaiveBayes, SMO and AdaBoostM1 are much lower than that of CSMRLP, suggesting that CSMRLP outperforms the other algorithms. More importantly, CSMRLP achieves higher F1 values without relying on training rate. This, again, shows the advantage of CSMRLP. Table 8 shows the Accuracy (bottom right) versus training rate in CSMRLP, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. In Table 8, we find that the Accuracy of CSMRLP is almost the highest among all the algorithms, and it is slightly lower than RandomForest only if the training rate is larger than 50 percent. The Accuracy of CSMRLP remains almost constant as training rate increases. Unlike CSMRLP, RandomForest is highly dependent on training rate, and the Accuracy of RandomForest significantly increases as training rate increases and it can achieve higher Accuracy only if the training rate is large enough. The accuracy values of the other algorithms do not change much as training rate increases. The above results demonstrate the robustness of the selected features and the algorithm CSMRLP constructed using them.

Table 9 reports the average value for Precision, Recall and F1 for question quality in MRLP, NaiveBayes, J48, RandomForest, SMO, AdaBoostM1, CSMRLP. From Table 9, we find that RandomForest has the highest average for Precision, and it is highly dependent on the training rate. However, CSMRLP has the second highest average for Precision, and it is not dependent on training rate. MRLP has the lowest average for Precision. This indicates that CSMRLP outperforms MRLP and almost all of the other classic classification

TABLE 9
Summary of the Results for QQ in Various Methods

	MRLP	NaiveBayes	J48	RandomForest	SMO	AdaBoostM1	CSMRLP
Precision	0.414	0.769	0.730	0.869	0.755	0.735	0.848
Recall	0.412	0.849	0.855	0.879	0.851	0.855	0.858
F1	0.413	0.791	0.787	0.853	0.788	0.789	0.848

algorithms on Precision. We also find that CSMRLP has the second highest average for Recall, and the average Recall value for CSMRLP is more than 0.857, which is much higher than MRLP. More importantly, it is not dependent on the training rate. This suggests that CSMRLP outperforms MRLP and most of the other classic algorithms on Recall. CSMRLP has the highest average for F1 than all the other algorithms except RandomForest and it does not depend on the training rate, while MRLP has the lowest average for F1. This shows that CSMRLP outperforms both MRLP and most of the other classic classification algorithms.

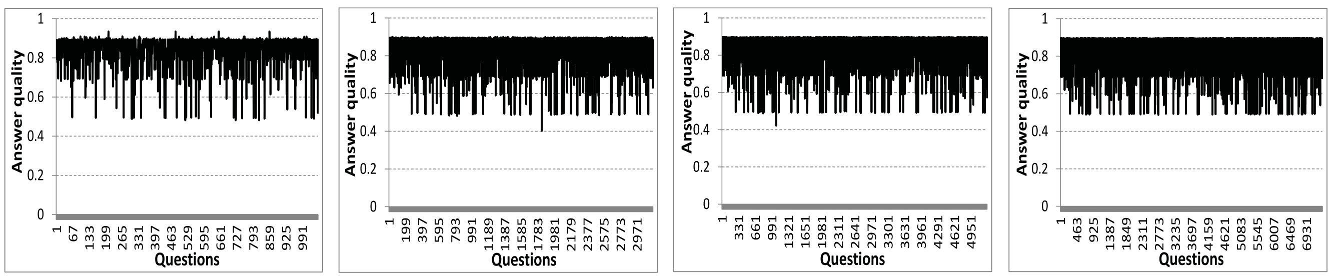
To estimate answer quality, we replace "Askers" with "Answerers" and utilize the same equations as before ((1)-(15)). Only in this case we replace Q (question quality) with A (answer quality) in equations (6), (7), (9), and (10) ((12) and (13) stay the same). With these equations we model the interaction between answerers and answers, the interaction between questions and answers, the interaction between categories and answers, and capture the mutual reinforcement relationship between answering expertise and AQ, the mutual reinforcement relationship between QQ and AQ, and the mutual reinforcement relationship between category reputation and AQ. Thus, another algorithm, CSMRLPAQ, is designed for answer quality estimation. Similar to the proof of convergence of CSMRLP in Theorem 5.2, the convergence of CSMRLPAQ can also be proved. Many systems allow users to place "like" or "dislike" tags to answers. The number of "likes" and "dislikes" is an indicator of the overall answer quality. In our future work, we will consider additional features (e.g., "like" or "dislike" tags) for evaluating AQ.

In order to fully verify the correctness of CSMRLPAQ, we randomly chose different number of questions as the testing samples (with the ratios of testing sample size to sample size lie in [10, 20 percent], [30, 40 percent], [50, 60 percent], [70, 80 percent], respectively), and estimated their answers' qualities using CSMRLPAQ and compared the result with the result of QQ estimation above. Fig. 7a shows the estimated answer quality for 1,054 questions randomly chosen from the sample questions. From the figure, we find that most questions can receive answers with a quality ≥ 0.6. A few questions have lower estimated answer quality, and this is because the questions are not well generated. Figs. 7b, 7c, and 7d show the estimated answer quality of 3,162, 5,271, and 7,379 questions randomly chosen from the sample questions, respectively. In these figures, we observe similar results.

That is, most questions can receive answers with a quality ≥ 0.6, with only a few questions having a lower estimated answer quality. The experimental result is consistent with the result of QQ estimation above, and it further confirms our conjecture: low-quality questions are likely to receive poor answers, while high-quality questions usually attract good answers. Question quality does determine answer quality to a certain extent.

The previous studies [3], [6] show that the QQ features influence the AQ. To better verify the correlation between QQ and AQ shown in Section 4.4.3, we tested if the features of high IG with respective to QQ are also important features for AQ. Fig. 8 shows the relationship between the average quality of the answers and the average weighted summation of features of top 20 IGs (WSFIG). WSFIG is the sum of the product of the top 20 IGs listed in Table 6 and normalized numerical values of the corresponding questions' features. Specifically, we used the formula $\prod_{i=1}^{20} IG_i \cdot F_i$ to calculate the WSFIG, where IG_i is the information gain of the question's ith feature in the feature list of top 20 IGs, and F_i is the normalized numerical value of the question's ith feature among the top 20 features. In the figure, we see that the quality of answers increases as WSFIG increases. This indicates that the features of high IG with respective to QQ are also important features for AQ, and thus verifies our analysis on the correlation between QQ and AQ in Section 4.4.3.

To verify the performance of CSMRLPAQ, we also separated the dataset into two parts: training data and testing data, and performed classification using CSMRLPAQ and five classical classification algorithms. For each algorithm, we also performed classification 10 times, and then used the mean value of each evaluation metric as the algorithm's performance on this metric. We also varied the training rate and performed classification multiple times for each algorithm at different training rates. Fig. 9 compares the performance of various evaluation metrics of CSMRLPAQ with those of other classic classification algorithms. Fig. 9a



(a) AQ of 1054 questions (b) AQ of 3162 questions (c) AQ of 5271 questions (d) AQ of 7379 questions

Fig. 7. Estimate answer quality of questions.

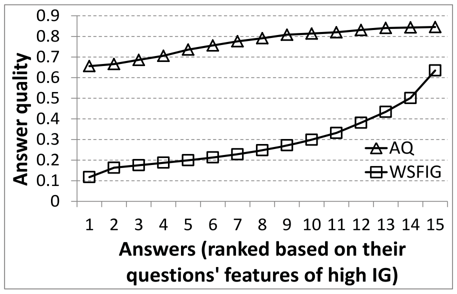


Fig. 8. The quality of answers versus their questions' features of high IG.

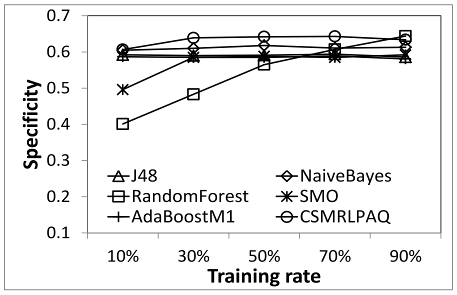
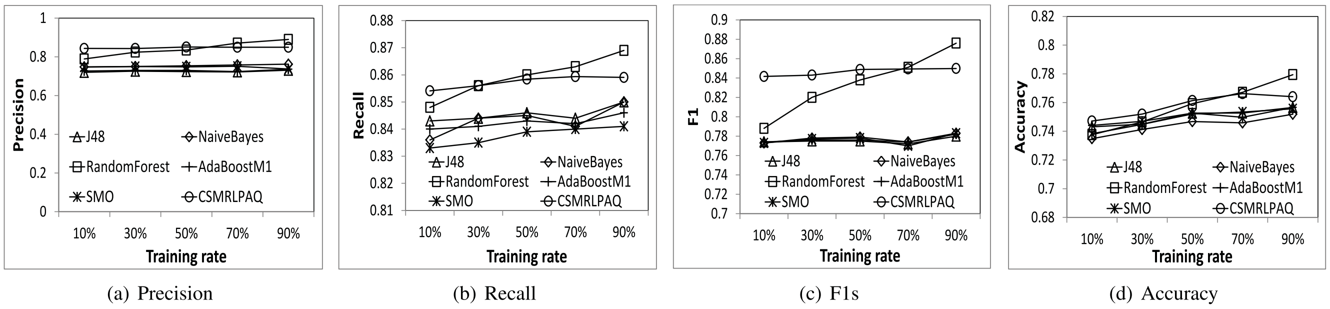


Fig. 10. Specificity in CSMRLPAQ and five other classic classification algorithms.

shows the Precision versus training rate using CSMRLPAQ, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. From Fig. 9a, we find that the Precision of CSMRLPAQ is higher than that of RandomForest when the training rate equals or less than 50 percent. After the training rate increases to 50 percent, the Precision of RandomForest is slightly higher than CSMRLPAQ. CSMRLPAQ has a Precision greater than 0.84. RandomForest, as a supervised method, highly relies on training rate [8], [20], and it can achieve higher Precision only if the training rate is large enough, which limits the applicability of RandomForest to new sites and domains. However, CSMRLPAQ is not dependent on training rate, and it can achieve high Precision at lower training rate. This is because CSMRLPAQ is a semi-supervised coupled mutual reinforcement framework, which can simultaneously calculate content quality (AQ and QQ), answering expertise, category reputation and thus requires relatively few labeled examples [6]. The Precisions of J48, NaiveBayes, SMO and AdaBoostM1 are lower than that of CSMRLPAQ. This demonstrates that CSMRLPAQ performs better than most of the classic classification algorithms. More importantly, it achieves higher Precision without relying on training rate, which overcomes the limitation of the supervised approaches. Fig. 9b shows the Recall versus training rate in CSMRLPAQ, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. In Fig. 9d, we find that the recall value of CSMRLPAQ is the highest among all the algorithms when the training rate is less than or equal to 30 percent. After the training rate increases to 30 percent, the recall value of CSMRLPAQ is slightly lower than RandomForest, but is still much higher than those of the other algorithms. The recall value of CSMRLPAQ stays constantly at around 0.857. Unlike CSMRLPAQ, RandomForest is highly dependent on training rate, and the recall value of RandomForest increases as the training rate increases. The recall values of the other algorithms are roughly the same value. This indicates CSMRLPAQ has a good performance on Recall, and it outperforms most of the other classic algorithms. Fig. 9c shows the F1 versus training rate in CSMRLPAQ, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. From Fig. 9c, we find that the F1 of CSMRLPAQ is the highest among all of the algorithms unless the training rate is larger than 70 percent. After the training rate increases to 70 percent, the F1 value of CSMRLPAQ is slightly lower than RandomForest, but it is still much higher than those of the other algorithms. The F1 of CSMRLPAQ is about 0.847 at different training rates, but the F1 of RandomForest is much lower than CSMRLPAQ when the training rate is less than 70 percent. RandomForest achieves higher F1 only if the training rate is large enough and thus it is highly dependent on training rate, but CSMRLPAQ is not dependent on training rate, and it can therefore achieve high F1 at lower training rate due to the same reason explained in Fig. 9a. The F1 values of J48, NaiveBayes, SMO and AdaBoostM1 are much lower than that of CSMRLPAQ, which suggests CSMRLPAQ outperforms the other algorithms. More importantly, CSMRLPAQ achieves higher F1 values without relying on training rate. This, again, indicates the advantage of CSMRLPAQ. Fig. 9d shows the Accuracy versus training rate in CSMRLPAQ, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1. In Fig. 9d, we find that the Accuracy of CSMRLPAQ is almost the highest among all the algorithms, and it is slightly lower than RandomForest only if the training rate is larger than 70 percent. The Accuracy of CSMRLPAQ remains almost constant as training rate increases. However, RandomForest is dependent on training rate, and the Accuracy of RandomForest increases as training rate increases and it can achieve higher Accuracy only if the training rate is large enough. The accuracy values of the other algorithms do not change much as training rate increases. The above results demonstrate the



(a) Precision (b) Recall (c) F1s (d) Accuracy

Fig. 9. Performance of various evaluation metrics for answer quality versus training rate in different methods.

TABLE 10
Summary of the Results for AQ in Various Methods

	NaiveBayes	J48	RandomForest	SMO	AdaBoostM1	CSMRLPAQ
Precision	0.754	0.725	0.842	0.747	0.729	0.847
Recall	0.843	0.854	0.859	0.838	0.842	0.857
F1	0.777	0.775	0.835	0.776	0.776	0.847

robustness of the selected features and the algorithm that CSMRLPAQ constructed using them.

Fig. 10 compares the Specificity of CSMRLPAQ with other classic classification algorithms. We find that the Specificity of CSMRLPAQ is higher than that of others, and it stays nearly constant at around 0.633 with a small fluctuation. The Specificity of RandomForest significantly increases as the training rate increases, whereas the Specificities of other algorithms do not change very much. NaiveBayes generates the second highest Specificity in all of the algorithms. CSMRLPAQ produces a higher Specificity (with a small fluctuation) than all of the other algorithms even when the class distribution is skewed and the positive class takes the largest proportion, which suggests that CSMRLPAQ performs better than the other algorithms.

Table 10 reports the average value for Precision, Recall and F1 for answer quality in CSMRLPAQ, J48, NaiveBayes, RandomForest, SMO, AdaBoostM1. From Table 10, we find that CSMRLPAQ has the highest average for Precision, and J48 has the lowest average for Precision. The average precision values of CSMRLPAQ, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1 follow: J48 < AdaBoostM1 < SMO < NaiveBayes < RandomForest < CSMRLPAQ. Also, CSMRLPAQ is not dependent on training rate, and it can achieve high Precision at lower training rate. This indicates that CSMRLPAQ outperforms all of the other classic classification algorithms on Precision. We also find that the average Recall value for CSMRLPAQ is more than 0.856, which is higher than all the other classic classification algorithms except RandomForest. This suggests that CSMRLPAQ outperforms most of the other classic classification algorithms on Recall. In addition, CSMRLPAQ has the highest average for F1, while J48 has the lowest average for F1. The average F1 values of CSMRLPAQ, J48, NaiveBayes, RandomForest, SMO and AdaBoostM1 follow: J48 < AdaBoostM1 = SMO < NaiveBayes < RandomForest < CSMRLPAQ. This shows that CSMRLPAQ outperforms the other classic classification algorithms.

7 Conclusion

In this paper, we analyze the factors affecting question quality and extract four kinds of features (i.e., asker-, category-, question- and answer-related) closely related to QQ. Then, we propose a CSMRLP algorithm to predict QQ based on these four kinds of features. We perform QQ classification and compare CSMRLP with a previously proposed MRLP algorithm and five other classic classification algorithms to verify the superior performance of our CSMRLP. To completely test the performance of CSMRLP, we vary training rate and perform classification multiple times, and test the performance using various evaluation metrics. In addition, we estimate the probability for a given question to receive a high quality answer. We also find the greatest factors in creating a high quality question; most notably the inclusion of "additional details", "category matching", and "Rep. rank of best answer". Finally, we provide suggestions on how to generate a question in order to achieve a high quality answer from the view of the question's features. In our future work, we will further study the influence of social features on the perceptions of question quality; we will take into account question type (including social questions) [34] to enhance question quality evaluation. we will also extract more features closely related to QQ and catch more characteristics of good quality questions and unanswered questions in our analysis. For the prediction task, we will further improve the prediction accuracy by performing multiclass classification. Considering answerers sometimes choose questions (especially opinion-related questions) to answer based on their interests, personal preferences and social closeness, a question's ability of attracting answerers' attention in this scenario may not accurately reflect the quality of the question. In our future work, we will additionally consider answerer-related features for question quality evaluation.

Acknowledgments

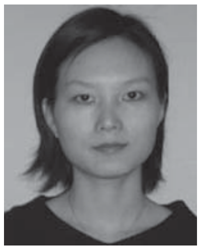
This research was supported in part by the US National Science Foundation (NSF) grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, and Microsoft Research Faculty Fellowship 8300751. Haiying Shen is the corresponding author.

References

[1] [Online]. Available: http://answers.yahoo.com, 2015.
[2] [Online]. Available: https://www.mturk.com/mturk/welcome, 2015.
[3] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne, "Finding high-quality content in social media," in *Proc. Int. Conf. Web Search Data Mining*, Palo Alto, CA, USA, 2008, pp. 183–194.
[4] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec, "Discovering value from community activity on focused question answering sites: A case study of stack overflow," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 850–858.
[5] D. Bazell and D. W. Aha, "Ensembles of classifiers for morphological galaxy classification," *Astrophysical J.*, vol. 548, pp. 219–223, 2001.
[6] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha, "Learning to recognize reliable users and content in social media with coupled mutual reinforcement," in *Proc. 18th Int. Conf. World Wide Web*, Madrid, Spain, 2009, pp. 51–60.
[7] M. Bouguessa, B. Dumoulin, and S. Wang, "Identifying authoritative actors in question-answering forums: The case of yahoo! answers," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Las Vegas, NV, USA, Aug. 2008, pp. 866–874.
[8] L. Breiman, "Random forests," *Mach. Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[9] X. Cao, G. Cong, B. Cui, and C. S. Jensen, "A generalized framework of exploring category information for question retrieval in community question answer archives," in *Proc. 19th Int. Conf. World Wide Web*, Raleigh, NC, USA, Apr. 2010, pp. 201–210.

[10] B. Caputo, E. Hayman, M. Fritz, and J.-O. Eklundh, "Classifying materials in the real world," *Image Vis. Comput.*, vol. 28, pp. 150–163, 2010.
[11] J. Cheng and R. Greiner, "Comparing Bayesian network class," in *Proc. 15th Conf. Uncertainty Artif. Intell.*, 1999, pp. 101–108.
[12] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bull.*, vol. 76, no. 5, pp. 378–382, 1971.
[13] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Mach. Learning*, 1996, pp. 148–156.
[14] F. M. Harper, D. Raban, S. Rafaeli, and J. A. Konstan, "Predictors of answer quality in online Q&A site," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, Florence, Italy, 2008, pp. 865–874.
[15] J. Jeon, W. Croft, J. Lee, and S. Park, "A framework to predict the quality of answers with Non-textual features," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2006, pp. 228–235.
[16] J. Jeon, W. B. Croft, and J. H. Lee, "Finding similar questions in large question and answer archives," in *Proc. 14th ACM Int. Conf. Inf. Knowl. Manag.*, 2005, pp. 84–90.
[17] G. Kazai, J. Kamps, M. Koolen, and N. Milic-Frayling, "Crowdsourcing for book search evaluation: Impact of hit design on comparative system ranking," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Beijing, China, Jul. 2011, pp. 205–214.
[18] V. Kitzie, E. Choi, and C. Shah, "Analyzing question quality through intersubjectivity: World views and objective assessments of questions on social question-answering," presented at the 76th ASIST Annu. Meeting: Beyond Cloud: Rethinking Information Boundaries, Montreal, QC, Canada, Nov. 2013.
[19] J. Ko, L. Si, and E. Nyberg, "A probabilistic framework for answer selection in question answering," in *Proc. NAACL/HLT*, 2007, pp. 524–531.
[20] H. Köpcke and E. Rahm, "Training selection for tuning entity matching," in *Proc. VLDB*, 2008, pp. 3–12.
[21] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proc. 10th Nat. Conf. Artif. Intell.*, 1992, pp. 223–228.
[22] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak, "Analyzing and predicting question quality in community question answering services," in *Proc. 21st Int. Conf. Companion World Wide Web*, 2012, pp. 775–782.
[23] B. Li and I. King, "Routing questions to appropriate answerers in community question answering services," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manag.*, 2010, pp. 1585–1588.
[24] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Sz, "Predicting web searcher satisfaction with existing Community-based answers," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Beijing, China, 2011, pp. 415–424.
[25] Y. Liu, J. Bian, and E. Agichtein, "Predicting information seeker satisfaction in community question answering," in *Proc. 31st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2008, pp. 483–490.
[26] G. Mountrakis, J. Im, and CaesarOgole, "Support vector machines in remote sensing: A review," *ISPRS J. Photogrammetry Remote Sensing*, vol. 66, pp. 247–259, 2011.
[27] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining Knowl. Discovery*, vol. 2, pp. 345–389, 1998.
[28] A. Pal, R. Farzan, J. A. Konstan, and R. E. Kraut, "Early detection of potential experts in question answering communities," in *Proc. 19th Int. Conf. User Modeling, Adaption, Personalization*, Berlin, Germany, 2011, pp. 231–242.
[29] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods.* Cambridge, MA, USA: MIT Press, 1999, pp. 185–208.
[30] J. Quinlan, "Improved use of continuous attributes in c4.5," *J. Artif. Intell. Res.*, vol. 4, no. 4, pp. 77–90, Jan. 1996.
[31] F. Riahi, Z. Zolaktaf, and M. Shafiei, "Finding expert users in community question answering," in *Proc. 21st Int. Conf. Companion World Wide Web*, 2012, pp. 791–798.
[32] E. Rodrigues and N. Milic-Frayling, "Socializing or knowledge sharing? characterizing social intent in community question answering," in *Proc. 18th ACM Conf. Inf. Knowl. Manag.*, Hong Kong, Nov. 2009, pp. 1127–1136.
[33] R. E. Schapire, Y. Singer, and A. Singhal, "Boosting and Rocchio applied to text filtering," in *Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 1998, pp. 215–223.
[34] C. Shah, V. Kitzie, and E. Choi, "Questioning the question—addressing the answerability of questions in community question-answering," in *Proc. Hawaii Int. Conf. Syst. Sci.*, Jan. 2014, pp. 1386–1395.
[35] H. Shen, Z. Li, J. Liu, and J. E. Grant, "Knowledge sharing in the online social network of yahoo! answers and its implications," *IEEE Trans. Comput.*, vol. 64, no. 6, pp. 1715–1728, Jun. 2015.
[36] A. Shtok, G. Dror, Y. Maarek, and I. Szpektor, "Learning from the past: Answering new questions with past answers," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 759–768.
[37] V. N. Vapn, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag, 1995.
[38] K. Wang, Z. Ming, and T.-S. Chua, "A syntactic tree matching approach to finding similar questions in Community-based QA services," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2009, pp. 187–194.
[39] X.-J. Wang, X. Tu, D. Feng, and L. Zhang, "Ranking community answers by modeling question-answer relationships via analogical reasoning," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2009, pp. 179–186.
[40] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen, "CQArank: Jointly model topics and expertise in community question answering," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manag.*, San Francisco, CA, USA, Nov. 2013, pp. 99–108.
[41] L. Zadeh, "From search engines to question answering systems—the problems of world knowledge, relevance, deduction and precisiation," in *Fuzzy Logic and the Semantic Web*, E. Sanchez, Ed. Amsterdam, The Netherlands: Elsevier, 2006, pp. 163-210.
[42] J. Zhang, M. Ackerman, and L. Adamic, "Expertise networks in online communities: Structure and algorithms," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 221–230.
[43] X. Zhu, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, USA, 2005.
[44] Z. Zhu, D. Bernhard, and I. Curevych, "A multi-dimensional model for assessing the quality of answers in social Q&A sites," Technische Universität Darmstadt, Darmstadt, Germany, Tech. Rep. TUD-CS-2009-0158, 2009.

Jinwei Liu received the MS degree in computer science from Clemson University, SC, and the University of Science and Technology of China, China, in 2012. He is currently working toward the PhD degree in the Department of Electrical and Computer Engineering, Clemson University. His research interests include wireless sensor networks, data mining, machine learning, social network, and cloud computing. He is a student member of the IEEE.



Haiying Shen (M'06-SM'13) received the BS degree in computer science and engineering from Tongji University, Shanghai, China, in 2000, and the MS and PhD degrees in computer engineering from Wayne State University, Detroit, MI, in 2004 and 2006, respectively. She is currently an associate professor with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She is a Microsoft Faculty fellow of 2010 and a member of the Association for Computing Machinery (ACM). She was the Program co-chair for a number of international conferences and member of the Program Committee of many leading conferences. She is a senior member of the IEEE.

Lei Yu received the PhD degree in computer science from Harbin Institute of Technology, China, in 2011. He was a postdoctoral research fellow in the Department of Electrical and Computer Engineering, Clemson University, SC. His research interests include sensor networks, wireless networks, cloud computing, and network security.