Web Service Personalized Quality of Service Prediction via Reputation-Based Matrix Factorization

Jianlong Xu, Zibin Zheng, *Member, IEEE*, and Michael R. Lyu, *Fellow, IEEE*

***Abstract*—With the fast development of Web services in service-oriented systems, the requirement of efficient Quality of Service (QoS) evaluation methods becomes strong. However, many QoS values are unknown in reality. Therefore, it is necessary to predict the unknown QoS values of Web services based on the obtainable QoS values. Generally, the QoS values of similar users are employed to make predictions for the current user. However, the QoS values may be contributed from unreliable users, leading to inaccuracy of the prediction results. To address this problem, we present a highly credible approach, called reputation-based Matrix Factorization (RMF), for predicting the unknown Web service QoS values. RMF first calculates the reputation of each user based on their contributed QoS values to quantify the credibility of users, and then takes the users' reputation into consideration for achieving more accurate QoS prediction. Reputation-based matrix factorization is applicable to the prediction of QoS data in the presence of unreliable user-provided QoS values. Extensive experiments are conducted with real-world Web service QoS data sets, and the experimental results show that our proposed approach outperforms other existing approaches.**

***Index Terms*—Matrix factorization, Quality of Service prediction, reputation, Web services.**

Acronyms and Abbreviations

QoS	Quality of Service
CF	collaborative filtering
MF	matrix factorization
RMF	reputation-based matrix factorization
NIMF	neighborhood integrated matrix factorization
HMF	hierarchical matrix factorization
EMF	extended matrix factorization
PMF	probabilistic matrix factorization
GDA	gradient decent algorithm
MAE	mean absolute error
RMSE	root mean squared error
IMean	item mean
UMean	user mean
PCC	person correlation coefficient
UPCC	user-based CF method using PCC
IPCC	item-based CF method using PCC
UIPCC	integrate UPCC and IPCC

Notations

d	a decay constant
e	the deviation
g	number of iterations in GDA
k	number of iterations in Algorithm 1
l	number of latent factors
m	number of users
n	number of Web services
q_{ij}	the i^{th} user's QoS value for service j
$\tilde{q}_{ij}$	the predicted value of q_{ij}
r_i	the ith user's reputation
r_i^k	r_i in the kth iteration
A_j	the average QoS value for service j
A_j^{k+1}	A_j in the $k+1$th iteration
H_j	the set of users which invokes service j
$\|H_j\|$	the number of users who invoke service j
I_{ij}	an indictor function
O_i	the set of services invoked by user i
$\|O_i\|$	the number of services invoked by user i
R	the set of users' reputation
RMaxI	the max iteration of calculating the reputation
Q	user-service matrix
$\tilde{Q}$	the approximate matrix of Q
QMaxI	the max iteration in GDA
S	Web services

Manuscript received September 01, 2014; revised February 03, 2015 and April 30, 2015; accepted July 29, 2015. Date of publication August 14, 2015; date of current version March 01, 2016. This work was supported in part by the National Basic Research Program of China (973 Project No. 2014CB347701), the National Natural Science Foundation of China (Project No. 61332010, 61472338), Guangdong Natural Science Foundation (Project No. 2014A030313151), and Research Grants Council of the Hong Kong Special Administrative Region, China (No. 415113). Associate Editor: W. E. Wong. (*Corresponding author: Zibin Zheng.*)

J. Xu and M. R. Lyu are with Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China.

Z. Zheng is with School of Advanced Computing, Sun Yat-sen University, and Collaborative Innovation Center of High Performance Computing, National University of Defense Technology, China (e-mail:zibin.gil@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2015.2464075

U	users
λ_U, λ_S	small positive decimals
$SumA$	summation of the product of q_{ij} and r_i^k
$SumR$	summation of the absolute value of $q_{ij} - A_j^k$
α	the learning rate
$\mathcal{L}$	the optimization function of MF
$\mathbb{R}^{l \times m}$	user and service latent feature matrices
$\|\cdot\|_F$	the Frobenius norm

I. Introduction

In modern society, complex software systems are becoming highly distributed, component-based, and service-oriented. These service-oriented systems are composed of a large number of software components discoverable at run-time, and run on a multitude of unknown, heterogeneous hardware and network platforms [1]. The distributed software components are usually implemented as Web services. A Web service is a software component designed to support interoperable machine-to-machine interaction over a network [2], which is becoming a major technique for building service-oriented systems [3], [4].

To make sure the service-oriented system remains reliable, efficient, and effective, service quality assurance techniques are needed [5], [6]. Quality of Service (QoS) is one of the most important aspects of software quality. In modern service-oriented systems, QoS includes a number of properties, such as response time, throughput, failure probability, availability, price, popularity, and so on [7]. In recent years, with the development of Internet technology, the QoS of service-oriented systems has become an important research topic to build high quality systems, and it has attracted great amounts of attention from both academia and industry [8]–[10]. In this paper, we focus on the QoS prediction of Web services as a new angle to attack the traditional issue of software quality assurance.

The prediction of QoS is an important means to build high quality service-oriented systems, as the predicted result can be employed by other service modeling and designing efforts to select more reliable service providers and better service provisioning mechanisms (for better networks, for example) for quality of service guarantees. In practice, values of the user-observed QoS properties (e.g., response time, throughput, failure probability) can vary widely for different users, influenced by unpredictable Internet connections, and heterogeneous user environments [11]. Because the QoS performance of Web services observed from the users' perspective is usually different from that declared by the service providers, personalized Web service QoS value prediction is becoming essential for designers of service-oriented systems to support service selection [12], service composition [13], fault-tolerant Web services [14], [15], and so on. In this paper, we focus on user side analysis (which emphasizes real-time user-observed QoS data) instead of server side analysis (which emphasizes the traditional testing and reliability data of servers).

Collaborative filtering (CF) is a widely used technique for Web service QoS prediction [16]–[18], which can be divided into memory-based CF, model-based CF, and other hybrid CF methods. Matrix factorization (MF) is a typical model-based approach in CF, which has been employed for QoS value prediction by academia and industry in recent years [19], [20]. In MF technology, the QoS values of Web services observed by different users can be represented as a user-service matrix. In the matrix, rows represent users, columns represent services, and each entry represents the QoS value of a service observed by a user. The main idea of MF-based QoS value prediction approaches is to train a model according to the available QoS values in the user-item matrix (i.e., historical QoS values contributed by different users) to predict missing QoS values in the user-item matrix [21]. Therefore, the reliability of user-contributed QoS values will highly influence the prediction accuracy of MF approaches. Unreliable users can cause negative impact on the prediction accuracy by providing unreliable QoS values.

Indeed, unreliable users have been found in many QoS prediction systems [22]. In these systems, some users may submit random or constant QoS values, while others (e.g., service providers) may give good QoS values for their own services, and bad QoS values for their competitors' services [23]. For example, suppose that, in a QoS prediction system, Service A and Service B provide the same functionality, and have very similar user-observed QoS performance by a user. In this condition, if this user submits a very high value to Service A, and a very low value to Service B, we can consider that the submitted QoS values are inconsistent. From the model's point of view, these two values are contradictory, and cannot be trusted. It is likely that this user is atypical, and provides unreliable QoS values. In previous studies of MF-based prediction approaches [20], [21], all users are treated equally, and their contributed values are assumed to be reliable. As a result, if unreliable users exist, the prediction accuracy is greatly affected by the unreliable QoS values.

It becomes an urgent task to explore a credible personalized prediction approach for efficient estimation of missing QoS values of Web services for different service users. To address this critical challenge, we present a reputation-based matrix factorization (RMF) for predicting the unknown Web service QoS values, which is different from traditional MF approaches. We incorporate the users' reputation into the Web service QoS prediction framework, and design a new algorithm to calculate the reputations of all users. Then MF-based predictions are conducted for missing QoS values by employing QoS values contributed by users and user reputations.

The contributions of this paper are summarized as follows. First, an algorithm is proposed to calculate the user reputation, and users are divided into different grades by their reputation. Second, we develop a highly credible approach for predicting unknown Web service QoS values, which combines user reputations and MF-based prediction methods, allowing the unknown QoS values to be predicted accurately. Finally, we conduct large-scale real-world experiments to study the prediction accuracy of our method compared with other state-of-the-art approaches.

The rest of this paper is organized as follows. Section II reviews the related work. Section III illustrates our QoS prediction framework. Section IV proposes a reputation-aware prediction algorithm. Section V presents experimental results, and Section VI concludes the paper.

II. Related Work

Web service QoS has been widely studied in service composition [24], and service selection [25]. Most of the previous research investigations assume that the service QoS values are obtained from third-party organizations or service providers easily. However, it is difficult to do so in reality. The QoS values may be unreliable, being influenced by the dynamic network environment. Moreover, some QoS properties (e.g., response time, failure rate, etc.) may vary over time among different users. And the QoS values we need may be unknown. Therefore, many researchers [18], [26], [27] have investigated how to predict the unknown QoS values.

There are mainly two types of approaches to predict QoS values for Web services. One is neighborhood-based (also named memory-based) collaborative filtering (CF) approaches, which utilize the historical invocation information of similar neighbors to make prediction. The neighborhood-based approaches are easy to understand and implement. However, they suffer from bad prediction accuracy when the data density is very low. Moreover, due to the similarity calculation, the time complexity of these approaches is all quadratic to the data size. Hence, they are not suitable to be used on very large datasets. Thus, more efficient models need to be explored, which intend to overcome the shortcomings listed above.

The other type of approach is model-based approaches. Matrix factorization (MF) is one of the most well-known model-based CF approaches, which is to exploit the latent factors that can determine QoS both from the user and the service aspects. As a typical model-based approach in collaborative filtering, matrix factorization models are accurate and scalable in many applications [28]–[30]. When employing MF to make QoS-predictions, a low-rank estimate to the original target matrix is desired. Zheng *et al.* [28] built a neighborhood integrated MF model named NIMF, which systematically fuses the neighborhood-based and model-based collaborative filtering approaches to achieve higher prediction accuracy. Lo *et al.* [29] proposed an extended matrix factorization (EMF) framework with relational regularization to make missing QoS value predictions, which aims to avoid the expensive and costly Web service invocations. To achieve higher QoS prediction accuracy, many researchers focus on integrating MF with additional information (e.g., geographical location, time, etc.). Zhang *et al.* [30] proposed a Web service QoS prediction framework called WSPred to provide time-aware personalized QoS value prediction for different service users, employing the tensor factorization technique to fit a factor model to the user-service-time tensor. The factorized user-specific, service-specific, and time-specific matrices are utilized to make comprehensive missing value prediction. Gantner *et al.* [31] used the mapping concept to construct an attribute-aware matrix factorization model for item recommendation from implicit and positive-only feedback. The factors of an MF model trained by standard techniques can be applied to the new-user and the new-item problems. He *et al.* [32] design a location-based hierarchical matrix factorization (HMF) method to perform personalized QoS prediction. The HMF model is trained in a hierarchical way by using the global QoS matrix, as well as several location-based local QoS matrices generated from user service clusters. Then the missing QoS values can be predicted by compactly combining the results from local matrix factorization and global matrix factorization. Although these approaches improve the prediction accuracy compared to the basic approaches, none of them take data credibility into consideration. In this paper, we employ users' reputation to improve the prediction accuracy over the traditional matrix factorization approach.

Reputation has been widely employed to measure the reliability of users in QoS prediction systems. The reputation score can be taken as a measurement of reliability for a user. Various reputation models for QoS prediction systems have been proposed. Tang *et al.* [33] proposed a hybrid trust-aware service recommendation method for a service-oriented environment with social networks via combining global trust and local trust evaluation. Qiu *et al.* [23] proposed a reputation-aware QoS value prediction approach based on CF, which first calculates the reputation of each user based on their contributed values, and then takes advantage of reputation-based ranking to exclude the values contributed by unreliable users. Finally, a neighborhood-based CF QoS prediction approach is used to predict the missing QoS values by employing information from reliable users and services. In this approach, the unreliable users are identified by calculating the reputation, and setting the threshold value. However, it is difficult to set up the threshold value, which may exclude the reliable users. In addition, the computational complexity of neighborhood-based methods employed in this paper is high. Based on the previous research investigations, we propose a new approach for Web service personalized QoS value prediction based on users' reputations and matrix factorization altogether.

III. The Prediction Framework

Previous studies of Web service QoS prediction frameworks such as EMF [29] and HMF [32] ignored the reliability of service users, and made prediction based on the collected QoS values from users directly. In fact, the QoS values may be unreliable, as they are provided by unreliable users. Therefore, the accuracy of prediction results is greatly affected by these unreliable QoS values. To address this critical challenge, we propose a highly credible reputation-based QoS prediction framework, as illustrated in Fig. 1.

As shown in the figure, users first submit their observed QoS values on the used Web services to our prediction server. Then, the QoS of the unused Web services can be predicted based on the submitted QoS values. The detailed steps are as follows.

1) Users invoke some remote Web services, and collect the user-observed QoS values of these services.
2) Users submit the collected QoS values to the QoS prediction server.
3) The server firstly collects the user-contributed QoS values, and fulfills the task of reputation computing based on the users' historic data, then evaluates the users based on the calculated user reputation, and finally makes personalized QoS prediction and return prediction results to the target user.
4) Users employ the corresponding prediction results to select optimal Web services to invoke.

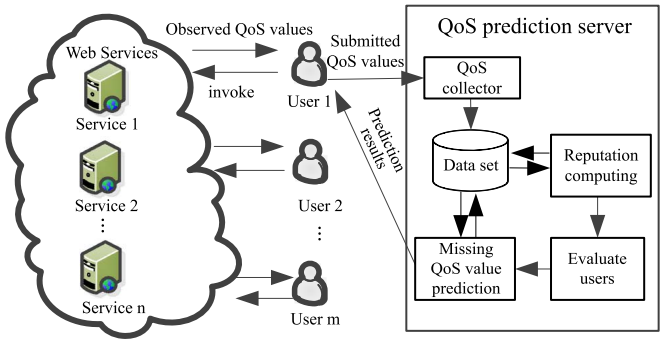

Fig. 1. Reputation-based QoS prediction framework.

IV. Reputation-Based Matrix Factorization

In this section, we first describe the mechanism of the basic MF model in Section IV-A. Then Section IV-B and Section IV-C present our RMF model in detail.

A. QoS Prediction With Basic Matrix Factorization

Generally, to predict missing QoS values, a factor model has to be fitted to the user-item matrix, and then use this factor model to make further predictions. Matrix factorization is a typical latent factor analysis model, which can factorize the high dimensional invocation matrix into two low dimensional feature matrices that are in the same feature space. In the two feature matrices, each column represents the user or service latent feature vector, which needs to be learned based on the known QoS records in the user-item matrix. With the help of statistical learning theory, all feature vectors are constructed separately. Once the stopping criterion is met, these feature spaces can recover all missing values in the original user-item matrix. The key step in the MF model is to build an objective function, under which two separate feature spaces can be reconstructed for higher prediction accuracy.

Given a set of m users $U = \{u_1, u_2, \ldots, u_m\}$, and a set of n Web services $S = \{s_1, s_2, \ldots, s_n\}$, the user-item matrix (each item is a Web service) is an $m \times n$ matrix Q. Each entry in this matrix q_{ij} ($i \leq m$, $j \leq n$) represents the value of a certain user-side QoS property (e.g., response time) of Web service j observed by service user i. If user i did not invoke Web service j before, then $q_{ij} = \text{null}$.

Let $U \in \mathbb{R}^{l \times m}$, and $S \in \mathbb{R}^{l \times n}$ represent user, and service latent feature matrices, respectively, where l is the number of latent factors. The number of factors l is called the dimensionality [27]. Usually, l is far less than m and n. Low-dimensional matrices U and S are unknown, and need to be estimated. In this factor model, a user's Web service QoS values correspond to a linear combination of the factor vectors, with user-specific coefficients. Furthermore, each column of U performs as a factor vector for a user, and each column of S is a linear predictor for a Web service. We can predict the entries in the corresponding column of the user-item matrix Q based on the factors in U and S Then, the invocation matrix Q can be factorized as the product of U and S approximately in the following equation.

$$Q \approx \tilde{Q} = U^T S, \tag{1}$$

where $\tilde{Q} = \tilde{q}_{ij} (1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n)$ is the approximate matrix of Q, in which $\tilde{q}_{ij}$ is the predicted value of q_{ij}. To reduce the total approximate errors, the difference between each pair of q_{ij} and $\tilde{q}_{ij}$ is to be minimized as

$$\min_{U,S} \mathcal{L}(U, S) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} \left(q_{ij} - U_i^T S_j \right)^2, \tag{2}$$

where U_i, and S_j are the ith, and jth column of U, and S respectively. I_{ij} is an indictor function, which indicates whether the QoS value on service j observed by user i in the matrix is missing. When I_{ij} is 1, that means q_{ij} is known, and 0 otherwise. To get rid of the over-fitting issue during the learning process, it is suggested to add two regularization terms to (2) [27], and the optimization problem is modeled as

$$\begin{aligned} &\min_{U,S} \mathcal{L}(U, S) \\ &= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} \left(q_{ij} - U_i^T S_l \right)^2 + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_S}{2} \|S\|_F^2, \end{aligned} \tag{3}$$

where $\| \cdot \|_F$ denotes the Frobenius norm, and λ_U, λ_S are both small positive decimals. Equation (3) is the objective function of the basic MF model, which minimizes the sum-of-squared-errors objective function with quadratic regularization terms. To get a local minimum of the objective function in (3), the gradient decent algorithm (GDA), an iterative algorithm, can be employed to reconstruct feature space U, and S by the following equations.

$$\begin{aligned} U_i' &= U_i - \alpha \frac{\partial \mathcal{L}}{\partial U_i}, \\ S_j' &= S_j - \alpha \frac{\partial \mathcal{L}}{\partial S_j}, \end{aligned} \tag{4}$$

where α is positive, and chosen as the learning rate. With the help of statistical learning theory, all feature vectors are constructed separately. Once the stopping criterion is met, these feature spaces can recover all missing values in the original matrix.

The key step of predicting QoS using the MF model is to build up an objective function, and factorize the invocation matrix into two feature matrices. In (3), the sum-of-squared-errors is related to the term $q_{ij} - U_i^T S_j$, which is affected by the reliability of the users' QoS values. Therefore, we have to take the users' reputation into consideration. In the following, we will calculate the user reputation, and then integrate it into the MF model.

B. Computation of User Reputation

Let the reputation of users be represented as $R = \{r_1, r_2, \ldots, r_m\} (0 \leqslant r_i \leqslant 1)$. When a user is reliable, the reputation value is 1. We employ the L1-AVG algorithm [23], [34] to calculate the reputation of a user, which takes an iterative approach to refine the users' reputation. The reputation r_i of user i is calculated as

$$\begin{cases} A_j^{k+1} = \frac{1}{|H_j|} \sum\limits_{u_i \in H_j} q_{ij} r_i^k \\ r_i^{k+1} = 1 - \frac{d}{|O_i|} \sum\limits_{s_j \in O_i} \left| q_{ij} - A_j^{k+1} \right|, \end{cases} \tag{5}$$

where k indicates the kth iteration, and r_i^k is the ith users reputation r_i in the kth iteration (the weight of different users). The computing method of (5) can be explained that a user's reputation depends on the gap between QoS values observed by this user and the average of QoS values observed by other users. If a user provides Web service QoS values which are quite different from the statistical mean of other users' QoS values, then the user is likely to be unreliable. Before calculating r_i^k, the kth iteration average QoS value A_j^k for service j must be calculated. A_j^k can be calculated by accumulating the product of q_{ij} and r_i^k, and then dividing by $|H_j|$, which denotes the number of users who have invoked service j. In (7), d is a decay constant in (0,1). It is used as a dumping factor, and to ensure that r_i^k is in (0,1). To reflect the deviation between q_{ij} and A_j^k, the absolute value of $q_{ij} - A_j^k$ is made. In the set of services invoked by user i (denoted as O_i), the accumulation of the absolute value is divided by the number of services which have been invoked by user i (denoted as $|O_i|$).

In (5), r_i is determined by q_{ij}, and the weighted average of other users' evaluation on the same service. The calculation of user reputation is iterative. In initialization, k $= 0$, and $r_i^0 = 1$, which indicates that each user is taken as reliable. The methodology of the computation of reputation score is in Algorithm 1.

Algorithm 1 Users' Reputation Computation Algorithm

Input: training matrix Q, decay constant d, H_j, O_i, RMaxI, threshold;

Output: users' reputation r_i;

initialize $k = 0$, $r_i^0 = 1$, $|H_j| = 0$, $|O_i| = 0$;
while $k <$ RMaxI do
  for ($j = 1$; $j <= n$; $j++$)
    $SumA = 0$; $CountA = 0$;
    for ($i = 1$; $i <=$ length(H_j); $i++$)
      if $q_{ij} > 0$
        $SumA = SumA + q_{ij} \times r_i^k$;
        $|H_j| = CountA++$;
      end if
    end for
    $A_j^{k+1} \leftarrow SumA/|H_j|$;
  end for
  for ($i = 1$; $i <= m$; $i++$)
    $SumR = 0$; $CountR = 0$;
    for ($j = 1$; $j <= length(O_i)$; $j++$)
      if $q_{ij} > 0$
        $SumR = SumR + absolute(q_{ij} - A_j^{k+1})$;
        $|O_i| = CountR++$;
      end if
    end for
    $r_i^{k+1} \leftarrow 1 - d \times SumR/|O_i|$;
  end for
if absolute $(r_i^{k+1} - r_i^k) <$ threshold
then break;
end while

In Algorithm 1, we first initialize the parameters. Then, the average QoS value for service j, and the reputation of the user i are calculated according to (5). We set two variables named $SumA$, and $SumR$, which denote the summations of the product of q_{ij} and r_i^k, and the absolute value of $q_{ij} - A_j^k$ of in (5), respectively.

Their initial values are set to zero. RMaxI is the max iteration in the process of calculating the reputation. While the number of iterations k is more than max iterations, or the absolute of $r_i^{k+1} - r_i^k$ is less than a threshold value (a minimum) which can determine the computational accuracy, the algorithm will break and output the users' reputation.

This method has several advantages: 1) it is convergent to a unique fixed vector, 2) it is robust to unreliable users, 3) it is easy to be implemented in practice, and 4) it is scalable to large data sets. In addition, it is quite robust, and has good time and space complexity [34]. With the users' reputation information, we can now design our reputation-based matrix factorization (RMF) model for QoS value prediction.

C. Reputation-Based Matrix Factorization Algorithm

When factorizing a QoS value, many researchers did not consider the user reputation (in other words, treated all users' reputations as the same). In our approach, the users' reputations are possibly unequal in the prediction process. Assume the user-item matrix is an $m \times n$ matrix Q, which is predicted by two low-rank matrices U, and S whose sizes are $l \times m$, and $l \times n$, respectively. For matrix U or S, columns represent how much corresponding latent features will affect QoS values on the user side or the service side. U, and S can be calculated by (6), and the missing QoS values are made by (1).

$$\min_{U,S} \mathcal{L}(U,S) = \frac{1}{2}\sum_{i=1}^{m} r_i \sum_{j=1}^{n} I_{ij}\left(q_{ij} - U_i^T S_j\right)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_S}{2}\|S\|_F^2 \quad (6)$$

Equation (6) is the objective function of the RMF model. Different from (3), (6) takes an extra parameter r_i, which can act as a weight to impact the accuracy of the prediction.

To illustrate the usefulness of applying the weighting parameter r_i, we analyze the deviation between (3) and (6) as

$$e = \frac{1}{2}\sum_{i=1}^{m}(1 - r_i)\sum_{j=1}^{n} I_{ij}\left(q_{ij} - U_i^T S_j\right)^2. \quad (7)$$

From (6) and (7), we can observe the following.

1) The user's reputation depends on the user's condition in (6), while all users' reputations are equal to 1 in (3). In fact, different users may have different reputations. Therefore, our model is more reasonable.
2) If r_i is large (close to 1), the value of $\sum_{j=1}^{n} I_{ij}(q_{ij} - U_i^T S_j)^2$ will be strengthened in (6), and the value of e will be in close proximity to zero in (7). This result means that the QoS values observed by users with high reputation are given more attention, and the predicted results based on high reputation users will be more accurate.
3) If r_i is small (close to 0), the value of $\sum_{j=1}^{n} I_{ij}(q_{ij} - U_i^T S_j)^2$ will be weakened. This result means that the QoS values observed by users with low reputations are given less attention, and the predicted results based on unreliable users will lead to large deviations.

To get a minimum of the objective function in (6), we apply the gradient descent algorithm to iteratively recover user and service latent space on both U_i and S_i. The gradients can be computed by

$$\frac{\partial \mathcal{L}}{\partial U_i} = \lambda_U U_i - r_i \sum_{j=1}^{n} I_{ij} \left(q_{ij} - U_i^T S_j\right) S_j,$$
$$\frac{\partial \mathcal{L}}{\partial S_j} = \lambda_S S_j - r_i \sum_{j=1}^{n} I_{ij} \left(q_{ij} - U_i^T S_j\right) U_i^T. \quad (8)$$

The RMF-based QoS prediction construction procedure is summarized in Algorithm 2.

Algorithm 2 RMF-based QoS prediction construction

Input: training matrix Q;

Output: U, S;

1: initialize U and S with small random numbers; $g = 0$;

2: calculate users' reputation from algorithm 1;

3: update $\partial \mathcal{L}/\partial U_i$;

4: update $\partial \mathcal{L}/\partial S_j$;

5: while $g <$ QMaxI do

6: for ($i = 1$; $i <= m$; $i++$)

7: $U_i' \leftarrow U_i - \alpha \partial \mathcal{L}/\partial U_i$;

8: end for

9: for ($j = 1$; $j <= n$; $j++$)

10: $S_j' \leftarrow S_j - \alpha \partial \mathcal{L}/\partial S_j$;

11: end for

12: end while

In Algorithm 2, we use the gradient descent algorithm to solve the optimization problem in (6). First, we initialize the matrices U and S with small random values. Then, the gradient descent algorithm iteratively updates the matrices U and S. QMaxI is the max iteration in the process of the gradient descent algorithm. The variable g is the iteration number in the gradient descent algorithm. The parameter α is the iteration step which is used to control the speed of iteration. $\partial \mathcal{L}/\partial U_i$ and $\partial \mathcal{L}/\partial S_j$ are given in (8).

The process stops when the minimum of the objective function given in (6) is reached. When g is less than the max iteration, the algorithm will continue. The algorithm finally outputs the predicted U and S, and the approximate matrix of Q can be obtained by (1).

V. Experiment

In this section, we conduct experiments to validate our RMF approaches, and compare the results with those from other methods. Our experiments are intended to 1) verify the rationality of our proposed theorems and corollaries, 2) discuss how the model parameters affect the prediction accuracy, and 3) compare our RMF approaches with other state-of-the-art methods.

A. Dataset Description

In this paper, the real-world dataset released by Zheng *et al.* [35] is used in our experiments. This dataset includes a 339 by 5825 matrix containing 339 service users, and 5825 real Web services. 1,974,675 response-time and throughput records are collected. The range of the response time property is 0 to 20 seconds (s), while the range of the throughput is 0 to 1000 kbps. In our experiments, we use the response time dataset, and add 20 unreliable service users to the dataset. The QoS values of the unreliable users are pseudo-randomly generated.

B. Metrics

In this paper, the mean absolute error (MAE) and root mean squared error (RMSE) are utilized to measure the difference between the predicted values and observed values. MAE, and RMSE are respectively defined as

$$MAE = \frac{\sum |q_{ij} - \tilde{q}_{ij}|}{N}, \quad (9)$$
$$RMSE = \sqrt{\frac{\sum (q_{ij} - \tilde{q}_{ij})^2}{N}}, \quad (10)$$

where q_{ij} denotes the known QoS value of service j observed by service user i, $\tilde{q}_{ij}$ is the predicted QoS value, and N is the number of predicted values. MAE gives equal weights to all the individual differences. But when large errors are particularly undesirable, RMSE is more useful, because it gives high weights to large errors [28].

C. Performance Comparison

To prove the effectiveness of our reputation-based matrix factorization method, we ran extensive experiments on state-of-the-art QoS prediction methods, and compared our method with

them. Several representative approaches are selected to compare with our models, including the following.

1) UMean (user mean)—In this method, a QoS value is predicted by calculating the mean of all the known QoS values of a user.
2) IMean (item mean)—In this method, a QoS value is predicted by calculating the mean of all the known QoS values of services (items) observed by different users.
3) UPCC—In this approach, a QoS value is predicted by utilizing the similarity between each two users using the Pearson Correlation Coefficient (PCC), and the historical invocation records of similar users [36].
4) IPCC—This approach is similar to UPCC, but instead of making use of similar users, it pays attention to similar services. Missing QoS values are estimated by utilizing the QoS values of similar services [37].
5) UIPCC—This approach integrates UPCC and IPCC into a unified model by aggregating their predicted results together, which takes advantage of both similar users and similar services [38].
6) PMF—In this method, two low-rank matrices are used to predict missing values in the user-service matrix [39].

In our experiments, matrix density is defined as the density of the training dataset. In this experiment, each QoS prediction method is run on 6 different matrices, whose densities are 5%, 10%, 15%, 20%, 25%, and 30% respectively. For example, a matrix density of 15% means that 15% of the entries in the matrix are used for predicting missing QoS values, while the remaining 85% are ones waiting to be predicted. We set the percentage of unreliable users =2.79% (10 unreliable users in all 359 users) for all methods. Additionally, for RMF and PMF, λ_U and λ_S are both set to 25, dimensionality is set to 10, and the number of iterations is set to 20. Refer to [18], and [32]; the decay constant d in (5) is set to 0.1. In the following experiments, we use the same experimental settings.

Table I shows the MAE and RMSE results of different methods with different densities from 5% to 30%. To compare to the existing approach in [23] which considers users' reputations, we have combined the reputation factors into UIPCC (named UIPCC+R in Table I). The experimental results show several important results.

1) Our RMF approach can achieve smaller MAE and RMSE values than other methods for response-times with different matrix densities, which indicates further higher accuracy than existing approaches, and verifies the effectiveness of our approach.
   Concretely, compared with UIPCC+R, RMF achieves 4.93% improvement in MAE, and 1.30% improvement in RMSE. Compared with the PMF model, the RMF achieves 4.89%, and 1.28% improvement in MAE, and RMSE, respectively.
2) Compared with UPCC, IPCC, UIPCC, and UIPCC+R, RMF achieve better prediction accuracy. That result occurs because RMF employs all the available information in the user-item matrix for making predictions, while the neighbor-based approaches only employ the information of similar neighbors (users or items) for making predictions.
3) Compared with PMF, RMF provides better prediction accuracy under different matrix density settings. That result occurs because RMF adds the users' reputation to the prediction model, and can reduce the impact of unreliable users, while PMF only employs the raw user-item matrix, and the accuracy of prediction is affected by unreliable users.
4) When the matrix density is small (e.g., 5%, 10%, etc.), relative to other methods, the effect of RMF prediction accuracy is not obvious (e.g., PMF can improve accuracy by 0.2% from PMF in MAE when the matrix density MD=5%, etc). While compared to UIPCC+R, the effect of RMF prediction accuracy is very obvious (e.g., PMF can improve accuracy by 10.2% from UIPCC+R in MAE when the matrix density MD$= 5\%$). When the matrix density is large (e.g., 25%, 30%), the effect of RMF prediction accuracy is more obvious (e.g., PMF can improve the accuracy by 2.73% from PMF in MAE when MD$= 30\%$). While compared to UIPCC+R, the effect of RMF prediction accuracy becomes less obvious (e.g., RMF can improve accuracy by 1.24% from UIPCC+R in MAE when MD$= 30\%$). This observation result indicates that, by considering user reputation, our PMF approach can achieve better prediction accuracy.

In the following experiments, we will investigate the impact of parameters on our models' performance, including λ_U and λ_S, the percentage of unreliable users, and the matrix density.

D. Impact of λ_U and λ_S

The parameters λ_U and λ_S control the proportion of the two regularization terms which are used to avoid over-fitting in (3) in the final predicted value. In this experiment, we assume $\lambda_U = \lambda_S$. If λ_U and λ_S are too large or too small, the prediction accuracy will be unsatisfactory. To understand the impact of λ_U and λ_S, we perform experiments on the response time values; and we set the dimensionality to 10, and the percentage of unreliable users to 2.79%.

Fig. 2 shows that the impacts of parameters λ_U and λ_S are on the prediction results. We make several observations.

1) When the matrix density is 5%, MAE first decreases, and reaches the minimal value when the values of λ_U and λ_S are at about 20, while RMSE increases monotonously.
2) When the matrix density is from 10% to 20%, MAE and RMSE first decrease, and reach an optimal value when λ_U and λ_S are at about 20 to 30.
3) When the matrix density is more than 25%, MAE and RMSE first decrease, and then go to a steady value; the optimal values are reached when λ_U and λ_S are more than 35. Therefore, the optimal values of λ_U and λ_S can be set according to the matrix density.

E. Impact of the Percentage of Unreliable Users

To study the impact of the percentage of unreliable users on the prediction accuracy, we set the number of unreliable users to 5, 10, 15, 20, and 80, so the percentage of unreliable users is 1.39%, 2.79%, 4.18%, 5.57%, and 19.1%, respectively. For other parameters, we vary the matrix density from 5% to 30%,

TABLE I
ACCURACY COMPARISON ON RESPONSE TIME (A SMALLER RMSE VALUE MEANS BETTER PERFORMANCE)

Methods	Matrix Density (MD)											
	Density = 5%		Density = 10%		Density = 15%		Density = 20%		Density = 25%		Density = 30%	
	MAE	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*
UMEAN	0.8769	1.8555	0.8764	1.856	0.8757	1.856	0.8752	1.8557	0.875	1.8557	0.8746	1.8554
IMEAN	0.9102	1.6558	0.8897	1.5976	0.8818	1.5779	0.8772	1.568	0.8746	1.5622	0.8733	1.5584
UPCC	0.6161	1.3951	0.539	1.3125	0.4964	1.2582	0.4715	1.222	0.4568	1.1986	0.4473	1.1816
IPCC	0.7042	1.4425	0.6375	1.3637	0.5655	1.3044	0.5239	1.2655	0.4974	1.2309	0.4732	1.1985
UIPCC	0.6114	1.3856	0.5382	1.305	0.4924	1.2477	0.4637	1.2019	0.4472	1.1757	0.4343	1.1493
UIPCC+R	0.6029	1.3874	0.5288	1.2957	0.4821	1.2299	0.4518	1.1874	0.4354	1.1542	0.4289	1.1296
PMF	0.5577	1.4289	0.5055	1.2808	0.4824	1.2193	0.4668	1.1854	0.4551	1.1641	0.448	1.1508
RMF	**0.5411**	**1.426**	**0.4877**	**1.2754**	**0.4594**	**1.206**	**0.4414**	**1.166**	**0.4278**	**1.1413**	**0.4182**	**1.1235**
Impro.vs UIPCC+R	10.2%	-0.028%	9.38%	2.03%	4.71%	2.39%	2.30%	1.80%	1.74%	1.11%	1.24%	0.53%
Impro.vs PMF	2.98%	0.20%	3.52%	0.422%	4.77%	1.09%	5.44%	1.64%	5.998%	1.96%	6.65%	2.37%

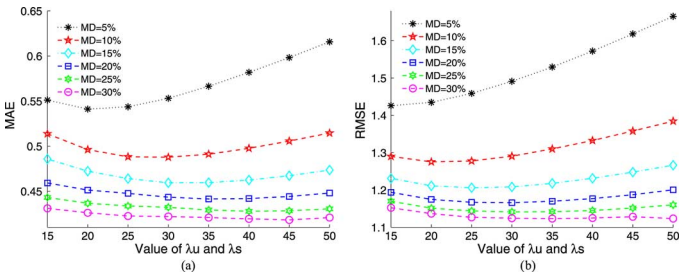

Fig. 2. Impact of λ_U and λ_S. (a) MAE; (b) RMSE.

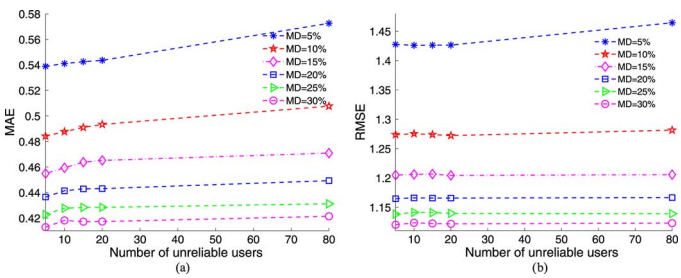

Fig. 3. Impact of the percentage of unreliable users. (a) MAE; (b)RMSE.

and the value of $\lambda_U = \lambda_S = 25$. The experimental results are illustrated in Fig. 3.

Fig. 3 shows that the prediction accuracy of RMF always keeps the best MAE and RMSE regardless of the change of the percentage of unreliable users in each matrix density. We can observe several results.

1) With the matrix density becoming denser, MAE and RMSE both become less. For different percentages of unreliable users, the denser matrix can provide more information, which is beneficial to improve the prediction accuracy.
2) When MD is smaller than 5%, the MAE or RMAE is more affected by the percentage of unreliable users (e.g., when MD is 5%, the MAE is 0.5389, 0.5436, and 0.5726, corresponding to the number of unreliable users of 5, 20, and 80, respectively).

When the matrix is dense (e.g., MD is more than 10%), the RMSE values are similar for different percentages of unreliable users. For example, when MD is 15%, RMSE values of 5, 20, and 80 unreliable users are 1.2737, 1.2722, and 1.2815, respectively. These RMSE values are similar to each other, because the impact of unreliable users is greatly reduced by our proposed approach. When MD is more than 10%, the changes of MAE and RMSE are minimal with the same matrix density, and different percentages of unreliable users (e.g., when MD is 15%, the RMSE is 1.2737, 1.2722, and 1.2815 corresponding to the number of unreliable users is 5, 20, and 80, respectively). The greater the matrix density, the smaller is the impact of the percentage of unreliable users. Therefore, the percentage of unreliable users has very small influence on the prediction when the range of the percentage of unreliable users is from 1.39% to 19.1%, because the unreliable users have low reputation in our RMF model, and the influence on the prediction results is reduced.

F. Impact of Matrix Density

Matrix density is the percentage of unremoved entries in the user-service matrix, which indicates how much available information we have to help us make predictions. To study the impact of matrix density, we vary the density of the matrix from 5% to 30% with a step value of 5%. We set the dimensionality to 10, select the number of unreliable users to be 5 and 20, and set $\lambda_U = \lambda_S = 25$, respectively. Fig. 4 illustrates the experimental results.

Fig. 4 shows that, with the increase of Matrix density, both MAE and RMSE decline rapidly at first. When the matrix density becomes larger, the speed of decrease slows down.

That means more accurate prediction results can be achieved by obtaining more QoS values.

G. Impact of Dimensionality

In our method, dimensionality denotes the number of latent features used to factorize the user-service matrix, which finally contributes to predicting the QoS value. Too small or large of a value will affect the prediction accuracy and efficiency. To study the impact of dimensionality, we tune the values of dimensionality from 1 to 40. For other parameters, we select the number of unreliable users to be 10, and set $\lambda_U = \lambda_S = 25$, respectively. Fig. 5 illustrates the experimental results.

Fig. 5 shows the impact of dimensionality on MAE, and RMSE, on our model, respectively. We can observe that both

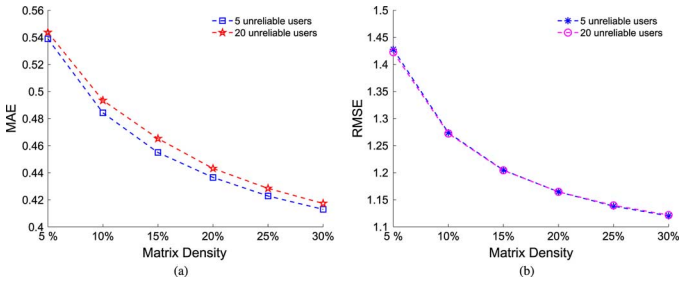

Fig. 4. Impact of matrix density. (a) MAE; (b) RMSE.

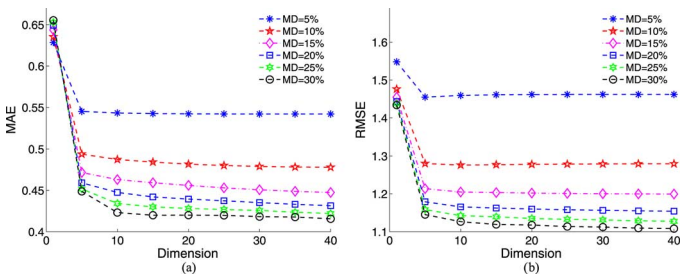

Fig. 5. Impact of dimensionality. (a) MAE; (b) RMSE.

MAE and RMSE are high at first, and decrease rapidly when the dimensionality is less than 10. When the dimensionality is larger than 10, both MAE and RMSE are low, and tend to be stable. To acquire good prediction results, we can raise the dimensionality. However, a larger dimensionality value will require longer computation time. In addition, if the dimensionality is set to a very high value, it will cause an overfitting problem, which will affect the prediction negatively [28].

VI. Conclusion

Users' reputation has great impact on the prediction of Web service QoS values. In this paper, we present an effective QoS prediction approach, namely RMF, for predicting unknown Web service QoS Values. We first calculate the reputation of each user based on their contributed QoS values. Then the user reputation is integrated into a matrix factorization (MF) prediction approach to get more accurate predictions. Extensive experiments are conducted on a real-world dataset, and the experimental results show that our proposed approach outperforms other existing approaches.

In the future, to continuously improve our prediction performance, we can try to optimize the reputation calculating method by employing intelligence algorithms to improve the calculation accuracy and rationality. Some feasible methods are listed as follows.

1) Exploiting statistical properties to measure users' consistency, and to identify unreliable users.
2) Taking the online condition into consideration, we may investigate reputation estimation methods (e.g. collaborative filtering method) to keep track of users' reputation, and detect spammers or unreliable users. We will also improve the RMF method to be suitable for an on-line algorithm.
3) Moreover, in terms of matrix factorization itself, we will attempt to composite other related information into our model to improve the prediction outcome, such as location, or time.

References

[1] M. Woodside, G. Franks, and D. C Petriu, "QoS management in service-oriented architectures," in *Proc. ACM Workshop oFuture of Software Engineering, FOSE 2007*, 2007, pp. 23–25, Univ. Carlton, Ottawa, ON, Canada.
[2] L. Zhang, J. Zhang, and H. Cai, *Services Computing: Core Enabling Technology of the Modern Services Industry*. Beijing, China: Tsinghua Univ. Press, 2007.
[3] Z. Zheng, "QoS management of web services," Ph.D. dissertation, The Chinese University of Hong Kong, Shenzhen, China, 2011.
[4] B. A. Malloy, N. A. Kraft, J. O. Hallstrom, and J. M. Voas, "Improving the predictable assembly of service-oriented architectures," *IEEE Softw.*, vol. 23, pp. 12–15, 2006.
[5] J. M. Voas and W. W. Agresti, "Software quality from a behavioral perspective," *IT Profession.*, vol. 6, pp. 46–50, 2004.
[6] V. Debroy and W. E. Wong, "A consensus-based strategy to improve the quality of fault localization," *Softw., Pract. Exper.*, vol. 43, pp. 989–1011, 2013.
[7] D. A. Menasce, "QoS issues in web services," *IEEE Internet Computi.*, vol. 6, pp. 72–75, 2002.
[8] J. El Haddad, M. Manouvrier, and M. Rukoz, "Tqos: Transactional and qos-aware selection algorithm for automatic web service composition," *IEEE Trans. Services Comput.*, p. 73C85, 2010.
[9] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized QoS-aware web service recommendation and visualization," *IEEE Trans. Services Comput.*, vol. 6, pp. 35–47, 2013.
[10] Z. ur Rehman, O. K. Hussain, and F. K. Hussain, "Parallel cloud service selection and ranking based on QoS history," *Int. J. Parallel Program.*, vol. 42, pp. 820–852, 2014.
[11] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Trans. Services Comput.*, vol. 7, pp. 32–39, 2012.
[12] L. Mei, W. K. Chan, and T. Tse, "An adaptive service selection approach to service composition," in *Proc. 6th In. Conf. Web Services (ICWS 2007)*, 2008, pp. 70–77.
[13] B. Q. Huang, C. H. Li, and F. Tao, "A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system," *Enterprise Inf. Syst.*, vol. 8, pp. 445–463, 2014.
[14] L. Yuan, J. S. Sun, J. Sun, and H. A. Basit, "Generic fault tolerant software architecture reasoning and customization," *IEEE Trans. Rel.*, vol. 55, pp. 421–435, 2006.
[15] N. Salatge and J.-C. Fabre, "Fault tolerance connectors for unreliable web services," in *Proc. 37th Int. Conf. Dependable Systems and Networks (DSN 2007)*, 2007, pp. 51–60.
[16] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. Services Comput.*, vol. 4, pp. 140–152, 2011.
[17] Y. Jiang, J. Liu, M. Tang, and X. Liu, "An effective Web service recommendation method based on personalized collaborative filtering," in *Proc. 9th Int. Conf. Web Services (ICWS 2011)*, 2011, pp. 211–218.
[18] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *Adv. Neural Inf. Process. Syst.*, vol. 20, pp. 1257–1264, 2008.
[19] Y. Xu, J. Yin, W. Lo, and Z. Wu, "Personalized location-aware QoS prediction for web services using probabilistic matrix factorization," in *Proc. 14th Int. Conf. Web Information System Engineering (WISE 2013)*, 2013, pp. 229–242.
[20] Y. Xu, J. Yin, and W. Lo, "A unified framework of QoS-based web service recommendation with neighborhood-extended matrix factorization," in *Proc. 6th IEEE Int. Conf. Service Oriented Computing and Applications (SOCA 2013)*, 2013, pp. 198–205.
[21] X. Luo, Y.-N. Xia, and Q.-S. Zhu, "Incremental collaborative filtering recommender based on regularized matrix factorization," *Knowl. Based Syst.*, vol. 27, pp. 271–280, 2012.
[22] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: A comprehensive survey," *Artif. Intell. Rev.*, p. 1C33, 2012.
[23] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu, "Reputation-aware QoS value prediction of web services," in *Proc. 10th IEEE Int. Conf. Services Computing (SCC 2013)*, 2013, pp. 41–48.
[24] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Softw. Eng.*, vol. 33, pp. 369–384, 2007.
[25] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Trans. Web (TWEB)*, vol. 1, no. 6, 2007.
[26] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, pp. 30–37, 2009.
[27] Z. Zheng and M. R. Lyu, "Personalized reliability prediction of web services," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 2, pp. 1–28, 2013.

[28] Z. Zheng, H. Ma, M. Lyu, and I. King, "Collaborative Web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Comput.*, vol. 6, pp. 289–299, 2013.
[29] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An extended matrix factorization approach for QoS prediction in service selection," in *Proc. 9th Int. Conf. Services Computing (SCC 2012)*, 2012, pp. 162–169.
[30] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A time-aware personalized QoS prediction framework for web services," in *Proc. 22th IEEE Symp. Software Reliability Engineering (ISSRE 2011)*, 2011, pp. 210–219.
[31] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations," in *Proc. IEEE 10th Int. Conf. Data Mining (ICDM 2010)*, 2010, pp. 176–185.
[32] P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu, "Location-based hierarchical matrix factorization for web service recommendation," in *Proc. IEEE 21st Int. Conf. Web Services (ICWS 2014)*, 2014, pp. 297–304.
[33] M. Tang, Y. Xu, J. Liu, Z. Zheng, and X. Liu, "Combining global and local trust for service recommendation," in *Proc. IEEE 21st Int. Conf. Web Services (ICWS 2014)*, 2014, pp. 305–312.
[34] R.-H. Li, J. X. Yu, X. Huang, and H. Cheng, "Robust reputation-based ranking on bipartite rating networks," in *Proc. 2012 SIAM Int. Conf. Data Mining (SDM 2012)*, 2012, pp. 612–623.
[35] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world web services," in *Proc. IEEE 17th Int. Conf. Web Services (ICWS 2010)*, 2010, p. 83C90.
[36] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of Netnews," in *Proc. ACM Conf. Computer Supported Cooperative Work (CSCW 1994)*, 1994, pp. 175–186.
[37] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item based collaborative filtering recommendation algorithms," in *Proc. 10th Int. World Wide Web Conf. (WWW 2001)*, 2001, pp. 285–295.
[38] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "WSRec: A collaborative filtering based web service recommender system," in *Proc. 16th Int. Conf. Web Services (ICWS 2009)*, 2009, pp. 437–444.
[39] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. 21st Annu. Conf. Neural Information Processing Systems (NIPS 2007)*, 2007, pp. 1257–1264.

Jianlong Xu received the Ph.D. degree from the South China University of Technology in 2013.

He is a postdoctoral fellow at Shenzhen Research Institute, the Chinese University of Hong Kong. His research interests include service computing, Internet of things, and distributed computing.

Dr. Xu is a member of ACM.

Zibin Zheng received the Ph.D. degree from the Chinese University of Hong Kong in 2011.

He is an associate professor at Sun Yat-sen University, China. His research interests include service computing, cloud computing, and software reliability engineering.

Dr. Zheng received ACM SIGSOFT Distinguished Paper Award at ICSE 2010, and Best Student Paper Award at ICWS 2010.

Michael R. Lyu received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, in 1981; the M.S. degree in computer engineering from the University of California, Santa Barbara, in 1985; and the Ph.D. degree in computer science from the University of California, Los Angeles, in 1988.

He is currently a professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong, China. He is also a Croucher senior research fellow. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, web technologies, multimedia information processing, and e-commerce systems.

Dr. Lyu is a fellow of both the IEEE and the AAAS for his contributions to software reliability engineering, and software fault tolerance.