Collaborative Web Service QoS Prediction via Neighborhood Integrated Matrix Factorization

Zibin Zheng, Hao Ma, Michael R. Lyu, Fellow, IEEE, and Irwin King, Senior Member, IEEE

Abstract—With the increasing presence and adoption of Web services on the World Wide Web, the demand of efficient Web service quality evaluation approaches is becoming unprecedentedly strong. To avoid the expensive and time-consuming Web service invocations, this paper proposes a collaborative Quality-of-Service (QoS) prediction approach for Web services by taking advantages of the past Web service usage experiences of service users. We first apply the concept of user-collaboration for the Web service QoS information sharing. Then, based on the collected QoS data, a neighborhood-integrated approach is designed for personalized Web service QoS value prediction. To validate our approach, large-scale real-world experiments are conducted, which include 1,974,675 Web service invocations from 339 service users on 5,825 real-world Web services. The comprehensive experimental studies show that our proposed approach achieves higher prediction accuracy than other approaches. The public release of our Web service QoS dataset provides valuable real-world data for future research.

Index Terms—Web service, QoS prediction, user-collaboration, matrix factorization

1 INTRODUCTION

Web services are self-described software applications designed to support interoperable machine-to-machine interaction over a network via standard interfaces and communication protocols [1]. Strongly promoted by the leading industrial companies, Web services have been widely employed in a lot of domains. Quality-of-Service (QoS) is usually employed to describe the non-functional characteristics of Web services. With the growing presence and adoption of Web services on the World Wide Web, QoS has become an important selling and differentiating point of the functionally equivalent Web services.

In the recent literature, a number of QoS-based approaches have been proposed for Web service composition [2], [3], [4], Web service selection [5], [6], [7], [8], fault-tolerant Web services [9], and so on. Accurate QoS values of Web services are required for these QoS-based approaches to work well. To address the fundamental problem of how to obtain the Web service QoS values, effective and efficient Web service QoS value obtaining approaches are urgently needed. The QoS values of Web services can be measured either at the server-side or at the client-side. QoS values measured at the server-side (e.g., price, popularity, etc.) are usually advertised by the service providers and identical for different users, while QoS values measured at the client-side (e.g., response-time, throughput, availability, etc.) can vary widely among users influenced by the unpredictable Internet connections and the heterogeneous user environments. To obtain accurate and personalized client-side Web service QoS values for different service users, client-side Web service evaluations [10], [11], [12], [13] are usually needed.

However, conducting real-world Web service evaluation at the client-side is difficult and sometimes even impossible. Because: (1) Web service invocations may be charged since the Web services are usually provided and hosted by other organizations. Even if the Web services are free, executing real-world Web service invocations for evaluation purposes consumes resources of service providers and imposes costs of service users. (2) It is time-consuming and impractical for service users to evaluate all the Web service candidates, since there are a lot of Web services in the Internet. (3) Service users are usually not experts on Web service evaluation and the common time-to-market constraints make in-depth evaluations of the target Web services difficult.

Without sufficient client-side evaluations, accurate Web service QoS values cannot be obtained. It is thus difficult for various QoS-based approaches, which employ these QoS values as input, to work well. To attack this critical challenge, we propose a neighborhood-integrated matrix factorization (NIMF) approach for collaborative and personalized Web service QoS value prediction. The idea is that client-side Web service QoS values of a service user can be predicted by taking advantage of the past Web service usage experiences of other service users.

To encourage QoS value sharing among service users (usually developers of service-oriented systems),

Index Terms—Web service, QoS prediction, user-collaboration, matrix factorization
a framework is proposed based on a key concept of Web 2.0, i.e., user-collaboration. In this framework, the users are encouraged to contribute their individually observed Web service QoS information to exchange for accurate and personalized Web service QoS prediction. Employing the Web service QoS values from different users, our neighborhood-integrated matrix factorization (NIMF) approach first finds out a set of similar users for the current users by calculating user similarities. Then, the NIMF approach employs both the local information of similar users and the global information of all available QoS values for fitting a factor model, and use this factor model to make personalized Web service QoS prediction. By the collaboration of different service users, the QoS values of a Web service can be effectively predicted in our approach even the current user did not conduct any evaluation on the Web service and has no idea on its internal design and implementation details.

Our approach remedies the shortcomings of previous evaluation approaches [10], [11], [13] by avoiding the expensive and time-consuming real-world Web service invocations. Complementary to various QoS-based approaches for Web services, which mainly focus on using the QoS values, this paper focuses on providing accurate and personalized QoS values for the service users.

The contributions of this paper are two-fold:

- Firstly, we propose a neighborhood-integrated matrix factorization (NIMF) approach for personalized Web service QoS value prediction. Our approach explores the past Web service usage experiences of service users by systematically fusing the neighborhood-based and the model-based collaborative filtering approaches to achieve higher prediction accuracy.
- Secondly, we conduct large-scale experiments and release a real-world Web service QoS dataset\(^1\) for future research. To the best of our knowledge, the scale of our released Web service QoS dataset (including 339 distributed service users and 5,825 real-world Web services as shown in Figure 1) is the largest in the field of service computing. Based on this dataset, extensive experimental investigations are conducted to study the QoS value prediction accuracy of our approach.

The rest of this paper is organized as follows: Section 2 presents our collaborative QoS framework and personalized QoS value prediction approach. Section 3 describes our experiments. Section 4 introduces related work and Section 5 concludes the paper.

2 COLLABORATIVE QoS PREDICTION

In this section, we first present a collaborative QoS framework for collecting QoS values from different users in Section 2.1. Then, based on the collected Web service QoS values, we describe the Web service QoS value prediction problem in Section 2.2, and propose a solution in Section 2.3 to Section 2.5.

2.1 Collaborative QoS Framework

Quality-of-Service (QoS) is usually employed for describing non-functional characteristics of Web services. While the server-side QoS values provide good indications of the server capacities, client-side QoS values provide more realistic measurements of the performance experienced by service users. Based on the previous investigations of Web service QoS [2], [3], [4], the commonly-used client-side Web service QoS properties include:

- **Response-time**: The time duration between a service user sending a request and receiving a response.
- **Throughput**: The average rate of successful message delivery over a communication channel. It is usually expressed as kilo bits per second (kbps).
- **Failure-probability**: The probability that a Web service invocation will fail.

To make accurate client-side QoS value predictions for a service user, our approach explores the past Web service usage experiences of different service users. Inspired by the recent success of YouTube\(^2\), Wikipedia\(^3\), and BitTorrent\(^4\), we apply user-collaboration, the key concept of Web 2.0, to collect Web service QoS values from different service users. As shown in Figure 2, the idea is that, instead of contributing videos (YouTube)
or knowledge (Wikipedia), the service users (usually developers of service-oriented systems) are encouraged to contribute/share their individually observed past Web service QoS information. In our framework, if a service user would like to obtain the QoS value prediction service from our centralized server, he/she needs to contribute some QoS values. The service users can provide the QoS values to our server via: (1) input the values to a Web form directly or upload a file following our format; (2) by running a client-side Web service evaluation application [14]; or (3) by employing a client-side middleware to automatically monitoring and contribute Web service QoS values [15], [16]. If a service user contributes more Web service QoS values, higher QoS value prediction accuracy can be achieved in our approach (technical details will be introduced in Section 2), since more user features can be mined from the contributed data. In this way, the service users are encouraged to contribute their observed Web service QoS values. Beside the user-contributed QoS values, we also control a set of distributed computers to monitor the QoS performance of real-world Web services (details will be introduced in Section 3.1). Based on the collected Web service QoS values, collaborative Web service QoS value prediction can be made.

### 2.2 Problem Description

The process of Web service QoS value prediction usually includes a user-item matrix as shown in Figure 3(a), where each entry in this matrix represents the value of a certain QoS property (e.g., response-time in this example) of a Web service (e.g., $i_j$ to $i_k$) observed by a service user (e.g., $u_1$ to $u_5$). As shown in Figure 3(a), each service user has several response-time values of their invoked Web services. Similarities between two different users in the matrix can be calculated by analyzing their QoS values on the same Web services. Pearson Correlation Coefficient (PCC) [17] is usually employed for the similarity computation. As shown in the similarity graph in Figure 3(b), totally 5 users (nodes $u_1$ to $u_5$) are connected with 10 edges. Each edge is associated with a PCC value in the range of $[-1, 1]$ to specify the similarity between user $u_i$ and user $u_j$, where larger PCC value stands for higher similarity. The symbol N/A means that the similarity between user $u_i$ and user $u_j$ is non-available, since they do not have any commonly invoked Web services. The problem we study in this paper is how to accurately predict the missing QoS values in the user-item matrix by employing the available QoS values. By predicting the Web service QoS values in the user-item matrix, we can provide personalized QoS value prediction on the unused Web services for the service users, who can employ these Web service QoS values for making service selection, service ranking, automatic service composition, etc.

To obtain the missing values in the user-item matrix, we can employ the Web service QoS values observed by other service users for predicting the Web service performance for the current user. However, since service users are in different geographic locations and are under different network conditions, the current user may not be able to experience similar QoS performance as other service users. To address this challenging Web service QoS value prediction problem, we propose a neighborhood-integrated matrix factorization (NIMF) approach, which makes the best utilization of both the local information of similar users and the global information of all the available QoS values in the user-item matrix to achieve better prediction accuracy. Our approach is designed as a two-phase process. In phase 1, we calculate the user similarities using PCC and determine a set of Top-K similar users for the current user. Then, based on the neighborhood information, we propose a neighborhood-integrated matrix factorization approach to predict the missing values in the user-item matrix in phase 2. Details of these two phases are presented at Section 3.2 and Section 3.3, respectively.

#### 2.3 Phase 1: Neighborhood Similarity Computation

Given an $m \times n$ user-item matrix $R$ consists of $m$ service users and $n$ Web services, each entry in this matrix $R_{ij}$ represents the value of a certain client-side QoS property of Web service $j$ observed by service user $i$. If user $i$ did not invoke the Web service $j$ before, then $R_{ij} = \text{null}$. Employing the available Web service QoS values in the user-item matrix, which are collected from different service
Consider an \( m \times n \) user-item matrix \( R \), the matrix factorization method employs a rank-\( l \) matrix \( X = UV^T \) to fit it, where \( U \in \mathbb{R}^{l \times m} \) and \( V \in \mathbb{R}^{l \times n} \). From the above definition, we can see that the low-dimensional matrices \( U \) and \( V \) are unknown, and need to be estimated. Moreover, this feature representations have clear physical meanings. In this linear factor model, a user’s Web service QoS values correspond to a linear combination of the factor vectors, with user-specific coefficients. More specifically, each column of \( U \) performs as a “factor vector” for a user, and each column of \( V \) is a linear predictor for a Web service, predicting the entries in the corresponding column of the user-item matrix \( R \) based on the “factors” in \( U \). The number of factors (in other word, the length of the “factor vector”) is called \textit{dimensionality}. By adding the constraints of the norms of \( U \) and \( V \) to penalize large values of \( U \) and \( V \), we have the following optimization problem [20]:

\[
\min_{U,V} \mathcal{L}(R, U, V) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^R (R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2, \tag{3}
\]

where \( I_{ij}^R \) is the indicator function that is equal to 1 if user \( u_i \) invoked Web service \( v_j \) and equal to 0 otherwise, \( \|U\|_F^2 \) denotes the Frobenius norm, and \( \lambda_U \) and \( \lambda_V \) are two parameters. The optimization problem in Eq. (3) minimizes the sum-of-squared-errors objective function with quadratic regularization terms. It also has a probabilistic interpretation with Gaussian observation noise, which is detailed in [20].

The above approach utilizes the global information of all the available QoS values in the user-item matrix for predicting missing values. This approach is generally effective at estimating overall structure (global information) that relates simultaneously to all users or items. However, this model are poor at detecting strong associations among a small set of closely related users or items (local information), precisely where the neighborhood models would perform better. Normally, the available Web service QoS values in the user-item matrix are very sparse; hence, neither of the matrix factorization or neighborhood-based approaches can generate optimal QoS values. In order to preserve both global information and local information mentioned above, we employ a balance parameter to fuse these two types of information. The idea is that every time when factorizing a QoS value, we treat it as the ensemble of a user’s information and the user’s neighbors’ information. The neighbors of the current user can be obtained by employing Eq. (2). Hence, we can minimize the following sum-of-squared-errors objective functions with quadratic regularization terms:
\[ \mathcal{L}(R, S, U, V) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^R (R_{ij} - (\alpha U_i^T V_j + (1 - \alpha) \sum_{k \in T(i)} S_{ik} U_k^T V_j))^2 \\
+ \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2, \]

where \( T(i) \) is a set of Top-K similar users of user \( u_i \) and \( S_{ik} \) is the normalized similarity score between user \( u_i \) and user \( u_k \), which can be calculated by:

\[ S_{ik} = \frac{PCC(i, k)}{\sum_{k \in T(i)} PCC(i, k)} \]

A local minimum of the objective function given by Eq. (4) can be found by performing gradient descent in \( U_i, V_j \):

\[ \frac{\partial \mathcal{L}}{\partial U_i} = \alpha \sum_{j=1}^{n} I_{ij}^R V_j ((\alpha U_i^T V_j + (1 - \alpha) \sum_{k \in T(i)} S_{ik} U_k^T V_j) - R_{ij}) \\
+ (1 - \alpha) \sum_{p \in B(i)} \sum_{j=1}^{n} I_{ij}^R S_{pi} V_j ((\alpha U_p^T V_j + (1 - \alpha) \sum_{k \in T(p)} S_{pk} U_k^T V_j) \\
- R_{pj}) + \lambda_U U_i, \]

\[ \frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^{m} I_{ij}^R ((\alpha U_i^T V_j + (1 - \alpha) \sum_{k \in T(i)} S_{ik} U_k^T V_j) - R_{ij}) \\
\times (\alpha U_i + (1 - \alpha) \sum_{k \in T(i)} S_{ik} U_k^T) + \lambda_V V_j, \]

where \( B(i) \) is the set that includes all the users who are the neighbors of user \( u_i \). In order to reduce the model complexity, in all of the experiments we conduct, we set \( \lambda_U = \lambda_V \).

### 2.5 Complexity Analysis

The main computation of the gradient methods is to evaluate the object function \( \mathcal{L} \) and its gradients against the variables. Because of the sparsity of matrices \( R \) and \( S \), the computational complexity of evaluating the object function \( \mathcal{L} \) is \( O(\rho_R l + \rho_R K l) \), where \( \rho_R \) is the number of nonzero entries in the matrix \( R \), and \( K \) is the number of similar neighbors. \( K \) is normally a small number since a large number of \( K \) will introduce noise, which will potentially hurt the prediction accuracy. The computational complexities for the gradients \( \frac{\partial \mathcal{L}}{\partial U} \) and \( \frac{\partial \mathcal{L}}{\partial V} \) in Eq. (7) are \( O(\rho_R K l + \rho_R K^2 l) \) and \( O(\rho_R l + \rho_R K l) \), respectively. Therefore, the total computational complexity in one iteration is \( O(\rho_R K l + \rho_R K^2 l) \), which indicates that theoretically, the computational time of our method is linear with respect to the number of observations in the user-item matrix \( R \). This complexity analysis shows that our proposed approach is very efficient and can scale to very large datasets.

### 3 Experiments

In this section, we conduct experiments to compare the prediction accuracy of our NIMF approach with other state-of-the-art collaborative filtering methods. Our experiments are intended to address the following questions: (1) How does our approach compare with the published state-of-the-art collaborative filtering algorithms? (2) How does the model parameter \( \alpha \) affect the prediction accuracy? (3) What is the impact of the matrix density, Top-K values, and dimensionality on the prediction accuracy?

#### 3.1 Dataset Description

We implement a WSCrawler and a WSEvaluator employing JDK 6.0, Eclipse 3.3, and Axis 2. Employing our WSCrawler, addresses of 5,825 openly-accessible Web services are obtained by crawling Web service information from www.seekda.com, a well-known Web service search engine. Axis2 is employed to generate client-side Web service invocation codes and test cases automatically. Totally 78,635 Java Classes and 13,644,507 lines of Java codes are generated in our experiments.

To evaluate the QoS performance of real-world Web services from distributed locations, we deploy our WSEvaluator to 339 distributed computers of PlanetLab\(^5\), which is a distributed test-bed made up of computers all over the world. In our experiment, each PlanetLab computer invokes all the Web services. As shown in Figure 1, totally 1,974,675 real-world Web service invocation results are collected from these 339 service users on 5,825 real-world Web services.

By processing the invocation results, we obtain two 339 x 5825 user-item matrices. One matrix contains

response-time values, while the other one contains throughput values. The statistics of our Web service QoS dataset is summarized in Table 1, the distributions of response-time and throughput values are shown in Figure 4, and more experimental details (e.g., detailed list of service users and Web services, the user-item matrix, the detailed Web service invocation results, etc.) are released online for future research. As shown in Table 1, the ranges of response-time and throughput are 0-20 seconds and 0-1000 kbps (kilo bits per second), respectively. Figure 4(a) shows that 91% of the response-time values are smaller than 2 seconds, and Figure 4(b) shows that 89.5% of the throughput values are smaller than 100 kbps.

Although we only study the response-time and throughput in the experiments, the proposed NIMF approach can be applied to other QoS properties easily. When predicting value of a certain QoS property, the value of the entry in the user-item matrix is the corresponding QoS value (e.g., response-time, throughput, failure probability, etc.) observed by a user on a certain Web service. Our NIMF approach can be employed on different QoS properties directly without any modifications.

3.2 Metrics

We use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics to measure the prediction quality of our method in comparison with other collaborative filtering methods. MAE is defined as:

\[
MAE = \frac{\sum_{i,j} |R_{ij} - \hat{R}_{ij}|}{N},
\]

where \( R_{ij} \) denotes the observed QoS value of Web service \( j \) observed by user \( i \), \( \hat{R}_{ij} \) is the predicted QoS value, and \( N \) is the number of predicted values. The MAE is the average over the verification sample of the absolute values of the differences between a prediction result and the corresponding observation. The MAE is a linear score which means that all the individual differences are weighted equally in the average.

RMSE is defined as:

\[
RMSE = \sqrt{\frac{\sum_{i,j} (R_{ij} - \hat{R}_{ij})^2}{N}}.
\]

In RMSE, the difference between a prediction result and the corresponding observed values are each squared and then averaged over the sample. Finally, the square root of the average is taken. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is most useful when large errors are particularly undesirable.

3.3 Comparison

In this section, in order to show the prediction accuracy of our NIMF approach, we compare our method with the following approaches.

1) UMEAN (User Mean): This method employs a service user’s average QoS value on the used Web services to predict the QoS values of the unused Web services.

2) IMEAN (Item Mean): This method employs the average QoS value of the Web service observed by other service users to predict the QoS value for a service user who never invoke this Web service previously.

3) UPCC (User-based collaborative filtering method using Pearson Correlation Coefficient): This method is a very classical method. In this paper, it employs similar users for the QoS value prediction [21], [22].

4) IPCC (Item-based collaborative filtering method using Pearson Correlation Coefficient): This method is widely used in industry company like Amazon. In this paper, it employs similar Web services (items) for the QoS value prediction [17].

5) UIPCC: This method combines the user-based and item-based collaborative filtering approaches and employs both the similar users and similar Web services for the QoS value prediction [23].

6) NMF (Non-negative Matrix Factorization): This method is proposed by Lee et al. in [24], [25]. It differs from other matrix factorization methods in that it enforces the constraint that the factorized factors must be non-negative. NMF is also widely used in collaborative filtering community.

7) PMF (Probabilistic Matrix Factorization): This method is proposed by Salakhutdinov and Mnih in [20]. It uses user-item matrix for the recommendations, and it is based on probabilistic matrix factorization.

In the real-world, the user-item matrices are usually very sparse since a service user usually only invokes a small number of Web services. In this paper, in order to conduct our experiments realistically, we randomly remove entries from the user-item matrix to make the matrix sparser with different density (i.e., 5%, 10%, 15%, and 20%). Matrix density 5%, for example, means that we randomly select 5% of the QoS entries to predict the remaining 95% of QoS entries. The original QoS values of the removed entries are used as the expected values to study the prediction accuracy. The above seven methods together with our NIMF method are employed for predicting the QoS values of the removed entries. The parameter settings of our NIMF method are \( \alpha = 0.4 \), Top-K=10, \( \lambda_U = \lambda_V = 0.001 \), and dimensionality=10 in the experiments. The experimental results are shown in Table 2, and the detailed investigations of parameter settings will be provided in Section 3.4 to Section 3.7.
From Table 2, we can observe that our NIMF approach obtains smaller MAE and RMSE values (indicating better prediction accuracy) consistently for both response-time and throughput with different matrix densities. The MAE and RMSE values of throughput in Table 2 are much larger than those of response-time, since the range of throughput is 0-1000 kbps, while the range of response-time is only 0-20 seconds. With the increase of matrix density from 5% to 20%, the MAE and RMSE values of our NIMF method become smaller, since denser matrix provides more information for the missing value prediction. Among all the prediction methods, our NIMF method generally achieves better performance on both MAE and RMSE, indicating that integrating the neighborhood information into matrix factorization model can achieve higher value prediction accuracy. These experimental results demonstrate that our interpretation on the formation of QoS values is realistic and reasonable.

### 3.4 Impact of Parameter $\alpha$

In our NIMF method, the parameter $\alpha$ controls how much our method relies on the users themselves and their similar users. If $\alpha=1$, we only employ the users’ own characteristics for making prediction. If $\alpha=0$, we predict the users’ QoS values purely by their similar users’ characteristics. In other cases, we fuse the users’ own characteristics with the neighborhood information to make prediction. We observe that optimal $\alpha$ values decrease (prediction accuracy increases) at first, but when $\alpha$ surpasses a certain threshold, the MAE and RMSE values increase (prediction accuracy decreases) with further increase of the value of $\alpha$. This phenomenon confirms the intuition that purely using the matrix factorization method or purely employing the neighborhood-based method cannot generate better QoS value prediction performance than fusing these two favors together.

From Figure 5(a) and Figure 5(b), when using user-item matrix with 10% density, we observe that our NIMF method achieves the best performance when $\alpha$ is around 0.3, while smaller values like $\alpha=0.1$ or larger values like $\alpha=0.7$ can potentially degrade the model performance. As shown in Table 2 and Figure 5, the optimal values of $\alpha$ is also around 0.3 for MAE and around 0.6 for RMSE. The optimal values of MAE and RMSE are different since MAR and RMSE are different metrics following different evaluation criteria. As the same with Figure 5(a) to Figure 5(d), the optimal $\alpha$ values of Figure 5(e) to Figure 5(h) are all between 0.3 to 0.6. This observation indicates that optimally combining the two methods can achieve better prediction accuracy than purely or heavily relying one kind of method, and this is why we use $\alpha=0.4$ as the default settings in other experiments. The same as Table 2, another observation from Figure 5 is that denser matrix provides better prediction accuracy.

### 3.5 Impact of Matrix Density

As shown in Table 2 and Figure 5, the prediction accuracy of our NIMF method is influenced by the matrix density. To study the impact of the matrix density on the prediction results, we change the matrix density from 2% to 20% with a step value of 2%. We set Top-K=10, dimensionality=10, and $\alpha=0.4$ in this experiment.

Figure 6 shows the experimental results, where Figure 6(a) and Figure 6(b) are the experimental results of response-time, and Figure 6(c) and Figure 6(d) are the experimental results of throughput. Figure 6 shows that when the matrix density is increased from 2% to 4%, the prediction accuracy of the NIMF method is significantly enhanced. With the further increase of matrix density, the speed of prediction accuracy enhancement slows down.

<table>
<thead>
<tr>
<th>QoS Properties</th>
<th>Methods</th>
<th>Matrix Density = 5%</th>
<th>Matrix Density = 10%</th>
<th>Matrix Density = 15%</th>
<th>Matrix Density = 20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response-time (0-20 s)</td>
<td>UMEAN</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td></td>
<td>IMEAN</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td></td>
<td>UPCC</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td></td>
<td>IPCC</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td></td>
<td>UIPCC</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td></td>
<td>PMF</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td></td>
<td>NIMF</td>
<td>0.5982 1.5331</td>
<td>0.7015 1.5813</td>
<td>0.6261 1.4078</td>
<td>0.5917 1.3268</td>
</tr>
<tr>
<td>Throughput (0-1000 kbps)</td>
<td>UMEAN</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
<tr>
<td></td>
<td>IMEAN</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
<tr>
<td></td>
<td>UPCC</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
<tr>
<td></td>
<td>IPCC</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
<tr>
<td></td>
<td>UIPCC</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
<tr>
<td></td>
<td>PMF</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
<tr>
<td></td>
<td>NIMF</td>
<td>54.0084 110.2821</td>
<td>53.8792 110.1751</td>
<td>53.7114 110.1708</td>
<td>53.7114 110.1708</td>
</tr>
</tbody>
</table>
Fig. 5. Impact of Parameter $\alpha$ (Dimensionality = 10)

Fig. 6. Impact of Matrix Density (Dimensionality = 10, $\alpha = 0.4$)

Fig. 7. Impact of Parameter Top-K (Dimensionality = 10, $\alpha = 0.4$)
This observation indicates that when the matrix is very sparse, the prediction accuracy can be greatly enhanced by collecting more QoS values to make the matrix denser.

### 3.6 Impact of Top-K

The Top-K value determines the number of similar users employed in our NIMF method. To study the impact of the Top-K values on the prediction results, we vary the values of Top-K from 10 to 50 with a step value of 10. We set dimensionality=10, α=0.4, and matrix density=10% in this experiment.

Figure 7(a) and Figure 7(b) show the MAE and RMSE results of response-time, while Figure 7(c) and Figure 7(d) show the MAE and RMSE results of throughput. Figure 7 shows that the MAE and RMSE values slightly increase (prediction accuracy decrease) when the Top-K value is increased from 10 to 50. This is because too large Top-K value will introduce noise (dissimilar users), which will potentially hurt the prediction accuracy. In all the four figures from Figure 7(a) to Figure 7(d), the Top-K value of 10 obtains the best prediction accuracy, and this is why we use Top-K=10 as the default experimental settings in other experiments.

### 3.7 Impact of Dimensionality

Dimensionality determines how many factors are used to factorize the user-item matrix. To study the impact of the dimensionality, we vary the values of dimensionality from 10 to 100 with a step value of 10. We set Top-K=10, α=0.4, and matrix density=10% in this experiment.

Figure 8(a) and Figure 8(b) show the experimental results of response-time, while Figure 8(c) and Figure 8(d) show the experimental results of throughput. As shown in Figure 8, the values of MAE and RMSE decrease when the dimensionality is increased from 10 to 100. These observed results coincide with the intuition that relative larger values of dimensions generate better recommendation results. However, the computational time of our NIMF approach is linear with respect to the value of dimensionality. Larger dimensionality value will require longer computation time. Moreover, the dimensionality cannot be set to a very high value since it will cause the overfitting problem, which will potentially hurt the recommendation quality.

### 4 Related Work and Discussion

Web services QoS has been widely discussed in a number of research investigations [26], [27], [28], [29], [30]. Zeng et al. [4] employ five generic QoS properties (i.e., execution price, execution duration, reliability, availability, and reputation) for dynamic Web service composition. Ardagna et al. [3] use five QoS properties (i.e., execution time, availability, price, reputation, and data quality) when making adaptive service composition in flexible processes. Alrifai et al. [31] propose an efficient service composition approach by considering both generic QoS properties and domain-specific QoS properties. In this paper, we focus on predicting the client-side QoS values for the service users.

The previous QoS-based Web service approaches (e.g., Web service composition [31], [3], [32], [4], Web service selection [5], [6], [7], [8], etc.) usually assume that Web service QoS values are already known or can be easily obtained from the service providers or third-party registries. This paper complements these QoS-based approaches by providing a collaborative QoS value prediction approach. The predicted QoS values provided by our approach can be employed by other QoS-based approaches in the field of service computing.

Usage experience of different users is a necessity to address the complex problems faced by the software engineering society. Meneely et al. [33] investigate the relationship between developer collaboration structure and software product reliability. Bird et al. [34] study latent subcommunities from the email social network of several open source projects. In this paper, we apply the concept of user-collaboration to enable Web service usage experience sharing between service users (usually developers of the service-oriented systems). Our NIMF approach takes advantage of the past Web service usage experience of service users to make Web service QoS value prediction for the current user.

Collaborative filtering methods are widely adopted in commercial recommender systems [35], [36], [17]. Two types of collaborative filtering approaches are widely studied: neighborhood-based (memory-based) and model-based. The most analyzed examples of neighborhood-based collaborative filtering include user-based approaches [21], [37], item-based approaches [38],
Moreover, as shown in Section 3.3, our NIMF approach studies on 5,825 real-world Web services in this paper. In contrast to previous work, we conduct large-scale experimental studies on only 100 Web services are studied. Compared with these approaches simply employ a movie rating dataset, i.e., MovieLens [17], for experimental studies, which is not convincing enough. Shao et al. [22] propose a user-based approach (i.e., UIPCC) for QoS value prediction by combining the UPCC and IPCC approaches. However, only 20 Web services are involved in the experiments. Zheng et al. [23] propose a neighborhood-based approach (i.e., UIPCC) for QoS value prediction by combining the UPCC and IPCC approaches. However, only 339 service users in heterogenous environments on 5,825 Web services. To the best of our knowledge, our dataset is the largest-scale Web service QoS dataset in the published work of service computing.

5 Conclusion and Future Work

Based on the intuition that a user’s Web service QoS usage experiences can be predicted by both the user’s own characteristics and the past usage experiences of other similar users, we propose a neighborhood-integrated matrix factorization approach for making personalized QoS value prediction. Our NIMF approach systematically fuses the neighborhood-based and model-based collaborative filtering approaches to achieve higher prediction accuracy. The extensive experimental analysis shows the effectiveness of our approach.

After obtaining the predicted QoS values on the unused Web services, most service users will make invocations to the selected Web services. The QoS values of these Web service invocations contain valuable information for improving the QoS prediction accuracy. We plan to design better incentive mechanisms and automatic approaches to enable the real-time sharing of these Web service usage experiences among service users. Moreover, we plan to apply our approach to the cloud computing environments, where the Web service QoS value collection becomes easier, since the user applications which invoke the Web services are usually deployed and running on the cloud.

The NIMF approach in this paper can only be employed to predict client-side QoS properties which have different values among users. We plan to conduct more studies to prediction QoS values of server-side QoS properties. In this paper, due to the lack of real-world datasets for conducting experiments, we only conduct experimental studies on response-time and throughput. We are currently collecting data on failure-probabilities of the real-world Web services. The effort requires long observation duration and sufficient Web service invocations for accurate measurement. More experimental studies on the failure-probability and other Web service QoS properties will be conducted in our future work.

Acknowledgments

The authors appreciate the reviewers for their extensive and informative comments for the improvement of this manuscript. The work described in this paper was fully supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 
CUHK 415410 and No. CUHK 413210), fully supported by the National Science Foundation of China under the Grant No. 61100078, and sponsored in part by the National Basic Research Program of China (973) under Grant No. 2011CB302600.

REFERENCES

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Zibin Zheng is an associate research fellow at Shenzhen Research Institute, The Chinese University of Hong Kong. He is also a guest research members at State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications), Beijing, China. He received his Ph.D. degree from The Chinese University of Hong Kong in 2011. He received ACM SIGSOFT Distinguished Paper Award at ICSE'2010, Best Student Paper Award at ICWS'2010, First Runner-up Award at Postgraduate Research Paper Competition, and IBM Ph.D. Fellowship Award 2010-2011. He served as program committee member of IEEE CLOUD'2009, CLOUDCOMPUTING'2010, CLOUDCOMPUTING'2011, SCC'2011, etc. His research interests include service computing, cloud computing, and software reliability engineering.

Hao Ma received his B.Eng. and M.Eng. degree in School of Information Science and Engineering from Central South University, in 2002 and 2005 respectively. He worked as a System Engineer in Intel Shanghai before he joined CUHK. He received his Ph.D. degree from the Chinese University of Hong Kong in 2010. He is currently a researcher at Microsoft Research Redmond. His research interests are in Information Retrieval, Data Mining, Machine Learning, Social Network Analysis and Recommender System.

Michael R. Lyu received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1981; the M.S. degree in computer engineering from 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, Hong Kong, China. He is also Director of the Video over Internet and Wireless (VIEW) Technologies Laboratory. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, Web technologies, multimedia information processing, and E-commerce systems. He has published over 400 refereed journal and conference papers in these areas. He has participated in more than 30 industrial projects and helped to develop many commercial systems and software tools. He was the editor of two book volumes: Software Fault Tolerance (New York: Wiley, 1995) and The Handbook of Software Reliability Engineering (New York: IEEE and New McGraw-Hill, 1996). Dr. Lyu received Best Paper Awards at ISSRE’98 and ISSRE’2003. Dr. Lyu initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the Program Chair for ISSRE’96 and General Chair for ISSRE’2001. He was also PRDC’99 Program Co-Chair, WWW’10 Program Co-Chair, SRDS’2005 Program Co-Chair, PRDC’2005 General Co-Chair, ICEBE’2007 Program Co-Chair, SCC’2010 Program Co-Chair, and DSN’2011 General Chair, and served in program committees for many other conferences including HASE, IICECS, IST, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WI. He has been frequently invited as a keynote or tutorial speaker to conferences and workshops in the U.S., Europe, and Asia. He has been on the Editorial Board of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON RELIABILITY, the Journal of Information Science and Engineering, and Software Testing, Verification & Reliability Journal. Dr. Lyu is an IEEE Fellow and an AAAS Fellow for his contributions to software reliability engineering and software fault tolerance. Dr. Lyu is also a Croucher Senior Research Fellow.

Irwin King received the BSc degree in Engineering and Applied Science from California Institute of Technology, Pasadena, in 1984. He received his MSc and PhD degree in Computer Science from the University of Southern California, Los Angeles, in 1988 and 1993 respectively. He joined the Chinese University of Hong Kong in 1993. He has served as the Director of the Electronic Information Engineering Program at the Chinese University and as the Head of the Centre for Multimedia Technology Research. He is also Director of the Video over Internet and Wireless (VIEW) Technologies Laboratory. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, Web technologies, multimedia information processing, and E-commerce systems. He has published over 400 refereed journal and conference papers in these areas. He has participated in more than 30 industrial projects and helped to develop many commercial systems and software tools. He was the editor of two book volumes: Software Fault Tolerance (New York: Wiley, 1995) and The Handbook of Software Reliability Engineering (New York: IEEE and New McGraw-Hill, 1996). Dr. Lyu received Best Paper Awards at ISSRE’98 and ISSRE’2003. Dr. Lyu initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the Program Chair for ISSRE’96 and General Chair for ISSRE’2001. He was also PRDC’99 Program Co-Chair, WWW’10 Program Co-Chair, SRDS’2005 Program Co-Chair, PRDC’2005 General Co-Chair, ICEBE’2007 Program Co-Chair, SCC’2010 Program Co-Chair, and DSN’2011 General Chair, and served in program committees for many other conferences including HASE, IICECS, IST, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WI. He has been frequently invited as a keynote or tutorial speaker to conferences and workshops in the U.S., Europe, and Asia. He has been on the Editorial Board of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON RELIABILITY, the Journal of Information Science and Engineering, and Software Testing, Verification & Reliability Journal. Dr. Lyu is an IEEE Fellow and an AAAS Fellow for his contributions to software reliability engineering and software fault tolerance. Dr. Lyu is also a Croucher Senior Research Fellow.