QoS-Aware Web Service Recommendation by Collaborative Filtering

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

Abstract—With increasing presence and adoption of Web services on the World Wide Web, Quality-of-Service (QoS) is becoming important for describing nonfunctional characteristics of Web services. In this paper, we present a collaborative filtering approach for predicting QoS values of Web services and making Web service recommendation by taking advantages of past usage experiences of service users. We first propose a user-collaborative mechanism for past Web service QoS information collection from different service users. Then, based on the collected QoS data, a collaborative filtering approach is designed to predict Web service QoS values. Finally, a prototype called *WSRec* is implemented by Java language and deployed to the Internet for conducting real-world experiments. To study the QoS value prediction accuracy of our approach, 1.5 millions Web service invocation results are collected from 150 service users in 24 countries on 100 real-world Web services in 22 countries. The experimental results show that our algorithm achieves better prediction accuracy than other approaches. Our Web service QoS data set is publicly released for future research.

Index Terms—Web service, collaborative filtering, QoS, service recommendation, service selection.

1 Introduction

Web services are software components designed to support interoperable machine-to-machine interaction over a network [35]. The increasing presence and adoption of Web services on the World Wide Web demand effective recommendation and selection techniques, which recommend the optimal Web services to a service users from a large number of available Web services.

With the number increasing of Web services, Quality-of-Service (QoS) is usually employed for describing nonfunctional characteristics of Web services [34]. Among different QoS properties of Web services, some properties are user independent and have identical values for different users (e.g., *price, popularity, availability*, etc.). The values of the user-independent QoS properties are usually offered by service providers or by third-party registries (e.g., UDDI). On the other hand, some QoS properties are user dependent and have different values for different users (e.g., *response time*, *invocation failure rate*, etc.). Obtaining values of the user-dependent QoS properties is a challenging task, since real-world Web service evaluation in the client side [7], [31], [36] is usually required for measuring performance of the user-dependent QoS properties of Web services. Client-side Web service evaluation requires real-world Web service invocations and encounters the following drawbacks:

- First, real-world Web service invocations impose costs for the service users and consume resources of the service providers. Some Web service invocations may even be charged.
- Second, there may exist too many Web service candidates to be evaluated and some suitable Web services may not be discovered and included in the evaluation list by the service users.
- Finally, most service users are not experts on Web service evaluation and the common time-to-market constraints limit an in-depth evaluation of the target Web services.

However, without sufficient client-side evaluation, accurate values of the user-dependent QoS properties cannot be obtained. Optimal Web service selection and recommendation are thus difficult to achieve. To attack this critical challenge, we propose a collaborative filtering based approach for making personalized QoS value prediction for the service users. Collaborative filtering [10] is the method which automatically predicts values of the current user by collecting information from other similar users or items. Well-known collaborative filtering methods include user-based approaches [2], [14], [32] and item-based approaches [8], [16], [24]. Due to their great successes in modeling characteristics of users and items, collaborative filtering techniques have been widely employed in famous commercial systems, such as Amazon,[1] Ebay,[2] etc. In this paper, we systematically combine the user-based approach and item-based approach for predicting the QoS values for the current user by employing historical Web service QoS data from other similar users and similar Web services. Similar service users are defined as the service users who have similar historical QoS experience on the same set of commonly invoked Web services with the current user.

- *The authors are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong, China. E-mail: {zbzheng, hma, lyu, king}@cse.cuhk.edu.hk.*

Manuscript received 14 Sept. 2009; revised 7 Feb. 2010; accepted 11 Feb. 2010; published online 14 Dec. 2010.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2009-09-0194.
Digital Object Identifier no. 10.1109/TSC.2010.52.

1. http://www.amazon.com.
2. http://www.half.ebay.com.

Different from the traditional Web service evaluation approaches [7], [31], [36], our approach predicts user-dependent QoS values of the target Web services without requiring real-world Web service invocations. The Web service QoS values obtained by our approach can be employed by other QoS driven approaches (e.g., Web service selection [33], [34], fault tolerant Web service [38], etc.).

The contribution of this paper is three-fold:

- First, we propose a user-collaborative mechanism for collecting historical QoS data of Web services from different service users.
- Second, we propose a Web service QoS value prediction approach by combining the traditional user-based and item-based collaborative filtering methods. Our approach requires no Web service invocations and can help service users discover suitable Web services by analyzing QoS information from similar users.
- Finally, we conduct a large-scale real-world experimental analysis for verifying our QoS prediction results. 100 real-world Web services in 22 countries are evaluated by 150 service users in 24 countries. 1.5 millions Web service invocations are executed by these service users and detailed experimental results are reported. To the best of our knowledge, the scale of our experiment is the largest among the published work of Web service QoS evaluation and prediction. Our real-world QoS data set has been released online[3] for promoting future research and making our experiments reproducible.

The rest of this paper is organized as follows: Section 2 introduces a user-collaborative QoS data collection mechanism. Section 3 presents the similarity computation method. Section 4 proposes a Web service QoS value prediction approach. Section 5 shows the implementation and experiments. Section 6 describes related work and Section 7 concludes the paper.

2 User-Collaborative QoS Collection

To make accurate QoS value prediction of Web services without real-world Web service invocations, we need to collect past Web service QoS information from other service users. However, it is difficult to collect Web service QoS information from different service users due to: 1) Web services are distributed over the Internet and are hosted by different organizations. 2) Service users are usually isolated from each other. 3) The current Web service architecture does not provide any mechanism for the Web service QoS information sharing.

Inspired by the recent success of *YouTube*[4] and *Wikipedia*,[5] we propose the concept of *user-collaboration* for the Web service QoS information sharing between service users. The idea is that, instead of contributing videos (*YouTube*) or knowledge (*Wikipedia*), the service users are encouraged to contribute their individually observed past Web service QoS data. Fig. 1 shows the procedures of our user-collaborative QoS data collection mechanism, which are introduced as follows:

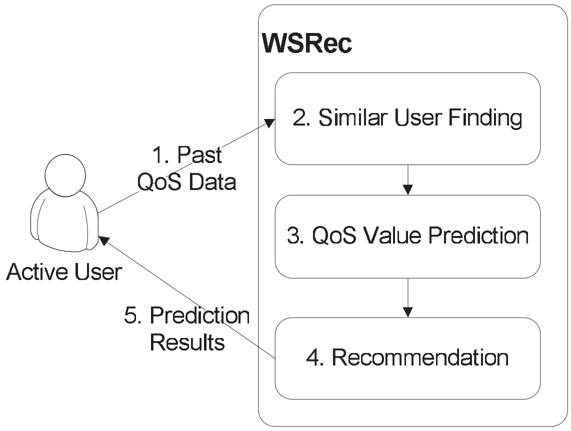

Fig. 1. Procedures of QoS Value Prediction.

1. A service user contributes past Web service QoS data to a centralized server WSRec [40]. In the following of this paper, the service users who require QoS value prediction services are named as *active users*.
2. WSRec selects similar users from the training users for the active user (technique details will be introduced in Section 3). *Training users* represent the service users whose QoS values are stored in the WSRec server and employed for making value predictions for the active users.
3. WSRec predicts QoS values of Web services for the active user (technique details will be introduced in Section 4).
4. WSRec makes Web service recommendation based on the predicted QoS values of different Web services (will be discussed in Section 4.4).
5. The service user receives the predicted QoS values as well as the recommendation results, which can be employed to assist decision making (e.g., service selection, composite service performance prediction, etc.).

In our user-collective mechanism, the active users who contribute more Web service QoS data will obtain more accurate QoS value predictions (details will be explained in Section 4). By this way, the service users are encouraged to contribute their past Web service QoS data. More architecture and implementation details of WSRec will be introduced in Section 5.1.

3 Similarity Computation

This section introduces the similarity computation method of different service users as well as different Web services (Step 2 of Fig. 1).

3.1 Pearson Correlation Coefficient

Given a recommender system consisting of M training users and N Web service items, the relationship between service users and Web service items is denoted by an $M \times N$ matrix, called the user-item matrix. Every entry in this

3. http://www.wsdream.net.
4. http://www.youtube.com.
5. http://www.wikipedia.org.

matrix $r_{u,i}$ represents a vector of QoS values (e.g., response time, failure rate, etc.) that is observed by the service user u on the Web service item i. If user u did not invoke the Web service item i before, then $r_{u,i} = null$. In the case that a Web service includes multiple operations, each item (column) of the user-item matrix represents a Web service operation instead of a Web service.

Pearson Correlation Coefficient (PCC) has been introduced in a number of recommender systems for similarity computation, since it can be easily implemented and can achieve high accuracy. In user-based collaborative filtering methods for Web services, PCC is employed to calculate the similarity between two service users a and u based on the Web service items they commonly invoked using the following equation:

$$Sim(a,u) = \frac{\sum_{i\in I}(r_{a,i}-\overline{r}_a)(r_{u,i}-\overline{r}_u)}{\sqrt{\sum_{i\in I}(r_{a,i}-\overline{r}_a)^2}\sqrt{\sum_{i\in I}(r_{u,i}-\overline{r}_u)^2}}, \qquad (1)$$

where $I = I_a \cap I_u$ is the subset of Web service items which both user a and user u have invoked previously, $r_{a,i}$ is a vector of QoS values of Web service item i observed by service user a, and $\overline{r}_a$ and $\overline{r}_u$ represent average QoS values of different Web services observed by service user a and u, respectively. From this definition, the similarity of two service users, $Sim(a,u)$, is in the interval of $[-1,1]$, where a larger PCC value indicates that service user a and u are more similar. When two service users have null Web service intersection ($I = null$), the value of $Sim(a,u)$ cannot be determined ($Sim(a,u) = null$), since we do not have information for the similarity computation.

Item-based collaborative filtering methods using PCC [8], [24] are similar to the user-based methods. The difference is that item-based methods employ the similarity between the Web service items instead of the service users. The similarity computation of two Web service items i and j can be calculated by

$$Sim(i,j) = \frac{\sum_{u\in U}(r_{u,i}-\overline{r}_i)(r_{u,j}-\overline{r}_j)}{\sqrt{\sum_{u\in U}(r_{u,i}-\overline{r}_i)^2}\sqrt{\sum_{u\in U}(r_{u,j}-\overline{r}_j)^2}}, \qquad (2)$$

where $Sim(i,j)$ is the similarity between Web service item i and j, $U = U_i \cap U_j$ is the subset of service users who have invoked both Web service item i and Web service item j previously, and $\overline{r}_i$ represents the average QoS values of the Web service item i observed by different service users. $Sim(i,j)$ is also in the interval of $[-1,1]$. When two Web service items have null service user intersection ($U = null$), the value of $Sim(i,j)$ cannot be computed ($Sim(i,j) = null$).

3.2 Significance Weighting

Although PCC can provide accurate similarity computation, it will overestimate the similarities of service users who are actually not similar but happen to have similar QoS experience on a few coinvoked Web services [19]. To address this problem, we employ a *significance weight* to reduce the influence of a small number of similar coinvoked items. An enhanced PCC for the similarity computation between different service users is defined as

$$Sim'(a,u) = \frac{2\times|I_a\cap I_u|}{|I_a|+|I_u|}Sim(a,u), \qquad (3)$$

where $Sim'(a,u)$ is the new similarity value, $|I_a \cap I_u|$ is the number of Web service items that are employed by both the two users, and $|I_a|$ and $|I_u|$ are the number of Web services invoked by user a and user u, respectively. When the coinvoked Web service number $|I_a \cap I_u|$ is small, the *significance weight* $\frac{2\times|I_a\cap I_u|}{|I_a|+|I_u|}$ will decrease the similarity estimation between the service users a and u. Since the value of $\frac{2\times|I_a\cap I_u|}{|I_a|+|I_u|}$ is between the interval of $[0,1]$ and the value $Sim(a,u)$ is in the interval of $[-1,1]$, the value of $Sim'(a,u)$ is in the interval of $[-1,1]$.

Just like the user-based methods, an enhanced PCC for the similarity computation between different Web service items is defined as

$$Sim'(i,j) = \frac{2\times|U_i\cap U_j|}{|U_i|+|U_j|}Sim(i,j), \qquad (4)$$

where $|U_i \cap U_j|$ is the number of service users who invoked both Web service item i and item j previously. Similar to $Sim'(a,u)$, the value of $Sim'(i,j)$ is also in the interval of $[-1,1]$.

As will be shown in our experimental results in Section 5.5, the *similarity weight* enhances the QoS value prediction accuracy of Web services. Based on the above similarity computation approach, if an active user provides more past QoS values of Web services to WSRec, the similarities computation will be more accurate, which will consequently improve the QoS value prediction accuracy. By this way, the service users are encouraged to provide more Web service QoS data.

4 QoS Value Prediction

In reality, the user-item matrix is usually very sparse [24], which will greatly influence the prediction accuracy. Predicting missing values for the user-item matrix can improve the prediction accuracy of active users [28]. Consequently, we propose a missing value prediction approach for making the matrix denser. The similar users or items of a missing value in the user-item matrix will be employed for predicting the value. By this approach, the user-item matrix becomes denser. This enhanced user-item matrix will be employed for the missing value prediction for the active users.

4.1 Similar Neighbors Selection

Before predicting the missing values in the user-item matrix, the similar neighbors of an entry, which include a set of similar users and a set of similar items, need to be identified. Similar neighbors selection is an important step for making accurate missing value prediction, since dissimilar neighbors will decrease the prediction accuracy. Traditional Top-K algorithms rank the neighbors based on their PCC similarities and select the top k most similar neighbors for making missing value prediction. In practice, some entries in the user-item matrix have limited similar neighbors or even do not have any neighbors. Traditional Top-K algorithms ignore this problem and still include dissimilar neighbors to predict the missing value, which will greatly reduce the prediction accuracy. To attack this problem, we propose an enhanced Top-K algorithm, where neighbors with PCC similarities smaller or equal to 0 will be excluded.

To predict a missing value $r_{u,i}$ in the user-item matrix, a set of similar users $S(u)$ can be found by the following equation:

$$S(u) = \{u_a | u_a \in T(u), Sim'(u_a, u) > 0, u_a \neq u\}, \tag{5}$$

and a set of similar Web service items $S(i)$ can be found by the following equation:

$$S(i) = \{i_k | i_k \in T(i), Sim'(i_k, i) > 0, i_k \neq i\}, \tag{6}$$

where $T(u)$ is a set of top k similar users to the user u and $T(i)$ is a set of top k similar items to the item i. By this way, the null intersection neighbors and the dissimilar neighbors with negative correlations will be discarded from the similar neighbor sets.

4.2 Missing Value Prediction

User-based collaborative filtering methods [2] (named as UPCC for ease of presentation) apply similar users to predict the missing QoS values by the following equation:

$$P(r_{u,i}) = \overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u)(r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)}, \tag{7}$$

where $P(r_{u,i})$ is a vector of predicted QoS values of the missing value $r_{u,i}$ in the user-item matrix, $\overline{u}$ is a vector of average QoS values of different Web services observed by the active user u, and $\overline{u}_a$ is a vector of average QoS values of different Web services observed by the similar service user u_a.

Similar to the user-based methods, item-based collaborative filtering methods [24] (named as IPCC) engage similar Web service items to predict the missing value by employing the following equation:

$$P(r_{u,i}) = \overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i)(r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)}, \tag{8}$$

where $P(r_{u,i})$ is a vector of predicted QoS values of the entry $r_{u,i}$ and $\overline{i}$ is a vector of average QoS values of Web service item i observed by different service users.

When a missing value does not have similar users, we use the similar items to predict the missing value, and vice versa. When $S(u) \neq \emptyset \wedge S(i) \neq \emptyset$, predicting the missing value only with user-based methods or item-based methods will potentially ignore valuable information that can make the prediction more accurate. In order to predict the missing value as accurate as possible, we systematically combine user-based and item-based methods to fully utilize the information of the similar users and similar items.

Since user-based method and item-based method may achieve different prediction accuracy, we employ two *confidence weights*, con_u and con_i, to balance the results from these two prediction methods. Confidence weights are calculated by considering the similarities of the similar neighbors. For example, assuming a missing value in the user-item matrix has three similar users with PCC similarity {1, 1, 1} and has three similar items with PCC similarity {0.1, 0.1, 0.1}. In this case, the prediction confidence by user-based method is much higher than the item-based method, since the similar users have higher similarities (PCC values) comparing with the similar items. Consequently, con_u is defined as

$$con_u = \sum_{u_a \in S(u)} \frac{Sim'(u_a, u)}{\sum_{u_a \in S(u)} Sim'(u_a, u)} \times Sim'(u_a, u) \tag{9}$$

and con_i is defined as

$$con_i = \sum_{i_k \in S(i)} \frac{Sim'(i_k, i)}{\sum_{i_k \in S(i)} Sim'(i_k, i)} \times Sim'(i_k, i), \tag{10}$$

where con_u and con_i are the prediction confidence of the user-based method and item-based method, respectively, and a higher value indicates a higher confidence on the predicted value $P(r_{u,i})$.

Since different data sets may inherit their own data distribution and correlation natures, a parameter λ ($0 \leq \lambda \leq 1$) is employed to determine how much our QoS value prediction approach relies on the user-based method and the item-based method. When $S(u) \neq \emptyset \wedge S(i) \neq \emptyset$, our method predicts the missing QoS value $r_{u,i}$ by employing the following equation:

$$\begin{aligned} P(r_{u,i}) = w_u \times &\left(\overline{u} + \frac{\sum_{u_a \in S(u)} Sim'(u_a, u)(r_{u_a,i} - \overline{u}_a)}{\sum_{u_a \in S(u)} Sim'(u_a, u)} \right) \\ + w_i \times &\left(\overline{i} + \frac{\sum_{i_k \in S(i)} Sim'(i_k, i)(r_{u,i_k} - \overline{i}_k)}{\sum_{i_k \in S(i)} Sim'(i_k, i)} \right), \end{aligned} \tag{11}$$

where w_u and w_i are the weights of the user-based method and the item-based method, respectively ($w_u + w_i = 1$). w_u is defined as

$$w_u = \frac{con_u \times \lambda}{con_u \times \lambda + con_i \times (1 - \lambda)} \tag{12}$$

and w_i is defined as

$$w_i = \frac{con_i \times (1 - \lambda)}{con_u \times \lambda + con_i \times (1 - \lambda)}, \tag{13}$$

where both w_u and w_i are the combinations of the *confidence weights* (con_u and con_i) and the parameter λ. The prediction confidence of the missing value $P(r_{u,i})$ by our approach using (11) can be calculated by equation

$$con = w_u \times con_u + w_i \times con_i. \tag{14}$$

When $S(u) \neq \emptyset \wedge S(i) = \emptyset$, since there are no similar items, the missing value prediction degrades to the user-based approach by employing (7), and the confidence of the predicted value is $con = con_u$. Similarly, when $S(u) = \emptyset \wedge S(i) \neq \emptyset$, the missing value prediction relies only on the similar items by employing (8), and the confidence of the predicted value is $con = con_i$. When $S(u) = \emptyset \wedge S(i) = \emptyset$, since there are no similar users or items for the missing value $r_{u,i}$, we do not predict the missing value in the user-item matrix. The prediction of $P(r_{u,i})$ is defined as

$$P(r_{u,i}) = null. \tag{15}$$

By the above design, instead of predicting all the missing values in the user-item training matrix, we only predict the missing values, which have similar users or similar items.

The consideration is that no prediction is better than bad prediction, since the user-item matrix will be involved for predicting QoS values for the active users and bad prediction will decrease the prediction accuracy for the active users. We also propose *confidence weights* (con_u and con_i) to balance the user-based prediction and the item-based prediction automatically. Moreover, a parameter λ is employed to enhance the feasibility of our method to different data sets. These designs are different from all other existing prediction methods and the experimental results in Section 5 show that these designs can significantly enhance the QoS value prediction accuracy of Web services.

4.3 Prediction for Active Users

After predicting missing values in the user-item matrix, we apply the matrix for predicting QoS values for active users. The prediction procedures are similar to the missing value prediction in Section 4.2. The only difference is that when $S(u) = \emptyset \wedge S(i) = \emptyset$, we predict the QoS values by employing the user-mean (UMEAN) and item-mean (IMEAN), where UMEAN is a vector of average QoS values of different Web services observed by the service user a and IMEAN is a vector of average QoS values of the Web service item i observed by different service users. The prediction formula is defined as

$$P(r_{a,i}) = w_u \times \overline{r}_a + w_i \times \overline{r}_i, \qquad (16)$$

where $\overline{r}_a$ is the UMEAN and $\overline{r}_i$ is the IMEAN. In this case, the confidence of the predicted value is $con = 0$.

4.4 Web Service Recommendation

After predicting the QoS values of Web services for an active user, the predicted QoS values can be employed by the following ways: 1) For a set of functionally equivalent Web services, the optimal one can be selected out based on their predicted QoS performance and the prediction confidence. 2) For the Web services with different functionalities, the top k best performing Web services can be recommended to the service users to help them discover potential good performing Web services. 3) The top k active service users, who have good predicted QoS values on a Web service, can be recommended to the corresponding service provider to help the provider find its potential customers.

Different from all other existing prediction methods, our method not only provides the predicted QoS values for the active users, but also includes the prediction confidences, which can be employed by the service users for better Web service selection.

4.5 Computational Complexity Analysis

This section discusses the upper bound on the worst-case computational complexity of the QoS value prediction algorithms. We assume there are m service users and n Web services in the training matrix.

4.5.1 Complexity of Similarity Computation

In Section 3, the computational complexity of $Sim(a, u)$ is $O(n)$, since there are at most n intersecting Web services between service user a and service user u. The computational complexity of $Sim(i, j)$ is $O(m)$, since there are at most m intersecting service users between Web service i and Web service j.

4.5.2 Complexity of UPCC

When predicting the missing values for an active user employing user-based PCC algorithm (7), we need to compute similarities of the active user with all the m training users in the training matrix (totally m similarity computations). As discussed in Section 4.5.1, the computational complexity of each similarity computation is $O(n)$. Therefore, the computational complexity of similarity computation is $O(mn)$.

The computational complexity of each missing value prediction for the active user is $O(m)$, since at most m similar users will be employed for the prediction. There are at most n missing values in an active user, so the computational complexity of the value prediction for an active user is $O(mn)$. Therefore, the total computational complexity of UPCC (including similarity computation and value prediction) is $O(mn)$.

4.5.3 Complexity of IPCC

When predicting the missing values for an active Web service employing item-based PCC algorithm (8), we need to compute similarities of the current Web service with all the n Web services in the training matrix (totally n similarity computations). As discussed in Section 4.5.1, the computational complexity of each similarity computation is $O(m)$. Therefore, the computational complexity of similarity computation is $O(mn)$.

After the similarity computation, for each missing value of an active Web service, the value prediction computational complexity is $O(n)$, since at most n similar Web services will be employed for the value prediction. There are at most m missing values in an active Web service, so the computational complexity of value prediction for an active Web service is $O(mn)$. Therefore, the same as UPCC, the computational complexity of IPCC is also $O(mn)$.

4.5.4 Complexity of Training Matrix Prediction

In Section 4.2, we predict the missing values in the training matrix. When employing UPCC approach, the computational complexity is $O(m^2 n)$ since there are at most m rows (users) to be predicted. When employing IPCC approach, the computational complexity is $O(mn^2)$ because there are at most n columns (Web services) to be predicted.

Since our approach is a linear combination of the UPCC and IPCC approaches, the computational complexity of our approach is $O(m^2 n + mn^2)$. Because the value prediction for the training matrix can be precomputed and recomputation is required only when the training matrix is updated, it will not influence the real-time prediction performance for active users.

4.5.5 Complexity of Active User Prediction

As discussed in Section 4.5.2, the computational complexity of UPCC for predicting values of an active user is $O(mn)$. When employing IPCC, the similarities of different columns (Web services) can be precomputed and there are at most n missing values in the active user. For the prediction of each missing value, the computational complexity is $O(n)$,

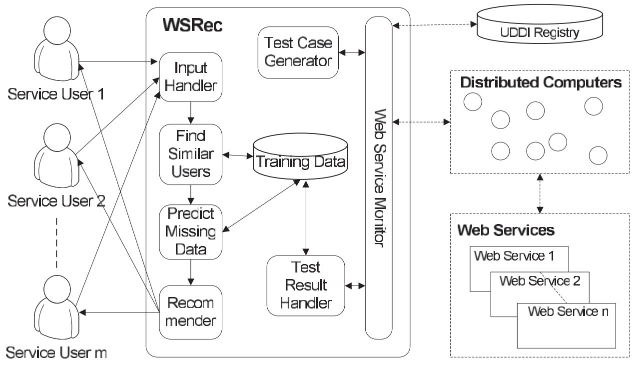

Fig. 2. Architecture of WSRec.

since at most n similar Web services will be employed for the prediction. Therefore, the computational complexity of IPCC for an active user is $O(n^2)$.

Since our QoS value prediction approach is a linear combination of UPCC and IPCC, the computational complexity of our approach for an active user is $O(mn + n^2)$.

5 Implementation and Experiments

5.1 Implementation

A prototype named *WSRec* [40] is implemented with JDK, Eclipse, Axis2,[6] and Apache Tomcat. In our prototype design, WSRec controls a number of distributed computers in different countries from Planet-lab[7] for monitoring the publicly available real-world Web services and collecting their QoS performance data. These collected real-world Web service QoS data are employed for studying the performance of our prediction approach. Fig. 2 shows the architecture of WSRec, which includes the following components

- The *Input Handler* receives and processes the Web service QoS values provided by an active service user.
- The *Find Similar Users* module finds similar users from the training users of WSRec for the active user.
- The *Predict Missing Data* module predicts the missing QoS values for the active user using our approach and saves the predicted values.
- The *Recommender* module employs the predicted QoS values to recommend optimal Web services to the active user. This module also returns all predicted values to the active user.
- The *Test Case Generator* generates test cases for the Web service evaluations. Axis2 is employed for generating test cases automatically in our implementation.
- The *Training Data* stores the collected Web service QoS values, which will be employed for predicting missing values of the active user.
- The *Test Result Handler* collects the Web service evaluation results from the distributed computers.
- The *Web Service Monitor* controls a set of distributed computers to monitor the Web services and record their QoS performance.

6. http://ws.apache.org/axis2.
7. http://www.planet-lab.org.

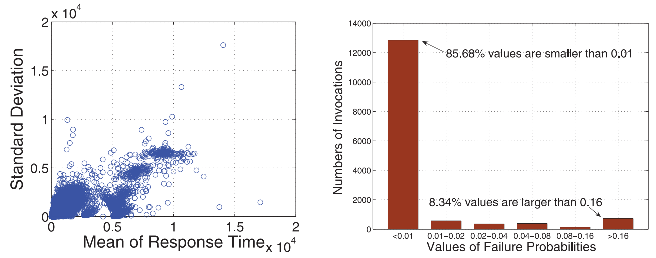

Fig. 3. Value distributions of the user-item matrix.

To obtain information of real-world Web services from the Internet, crawling programs are implemented. Totally 21,197 publicly available Web services are obtained by crawling Web service information from: 1) Well-known companies (e.g., *Google, Amazon, etc.*), 2) portal Websites that list publicly available Web services (e.g., *xmethods.net*, *webservicex.net*, *etc.*), and 3) Web service searching engines (e.g., *seekda.com*, *esynaps.com*, *etc.*). We successfully generate client stub classes for 18,102 Web services using the *WSDL2Java* tool from the *Axis2* package. A total of 343,917 Java Classes are generated. The Web services which fail during the client stub generation are mainly due to *network connection problems* (e.g., connection timeout, HTTP 400, 401, 403, 500, 502, and 503), *FileNotFoundException* and *InvalidWSDLFiles*.

Since it is difficult to monitor all the Web services at the same time, we randomly select 100 Web services which are located in 22 countries for our experiments. Some of the initially selected Web services have to be replaced due to: 1) authentication required, 2) permanent invocation failure (e.g., the Web service is shutdown), and 3) too long processing duration. One hundred and fifty computers in 24 countries from Planet-Lab [6] are employed to monitor and collect QoS information on the selected Web services. About 1.5 millions Web service invocations are executed and the test results are collected.

By processing the experimental results, we obtain a 150×100 user-item matrix, where each entry in the matrix is a vector including two QoS values, i.e., *response time* and *failure rate*. *Response time* represents the time duration between the client sending a request and receiving a response, while *failure rate* represents the ratio between the number of invocation failures and the total number of invocations. In our experiments, each service user invokes each Web service for 100 times. Figs. 3a and 3b show the value distributions of *response time* and *failure rate* of the 15,000 entries in the matrix, respectively. Fig. 3a shows that the means of response times of most entries are smaller than 5,000 milliseconds and different Web service invocations contain large variances in real environment. Fig. 3b shows that failure probabilities of most entries (85.68 percent) are smaller than 1 percent, while failure probabilities of a small part of entries (8.34 percent) are larger than 16 percent.

In the following sections, the unit of response time is milliseconds. Comprehensive analysis of the experimental results will be reported and more experimental raw data (e.g., the distributed computer nodes and Web services, the QoS user-item matrix, all the 1.5 millions invocation results, etc.) are provided online.[8]

8. http://www.wsdream.net.

TABLE 1
Experimental Parameter Descriptions

Symbols	Descriptions
Given number	the number of QoS values provided by an active user
Density	the density of the training matrix
Training users	the number of training users
Top-K	the number of similar neighbors employed for the value prediction
λ	determines how much our approach relies on the user-based approach or item-based approach

5.2 Experimental Setup

We divide the 150 service users into two parts, one part as training users and the other part as active users. For the training matrix, we randomly remove entries to make the matrix sparser with different density (e.g., 10, 20 percent, etc.). For an active user, we also randomly remove different number of entries and name the number of remaining entries as *given number*, which denotes the number of entries (QoS values) provided by the active user. Different methods are employed for predicting the QoS values of the removed entries. The original values of the removed entries are used as the expected values to study the prediction accuracy. The experimental parameters and their descriptions are summarized in Table 1.

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

$$MAE = \frac{\sum_{i,j} |r_{i,j} - \widehat{r}_{i,j}|}{N} \tag{17}$$

and RMSE is defined as

$$RMSE = \sqrt{\frac{\sum_{i,j} (r_{i,j} - \widehat{r}_{i,j})^2}{N}}, \tag{18}$$

where $r_{i,j}$ denotes the expected QoS value of Web service j observed by user i, $\widehat{r}_{i,j}$ is the predicted QoS value, and N is the number of predicted values.

5.3 Performance Comparison

To study the prediction performance, we compare our approach (named as *WSRec*) with four other well-known approaches: user-mean, item-mean, user-based prediction algorithm using PCC (*UPCC*) [2], and item-based algorithm using PCC (*IPCC*) [24]. UMEAN employs the average QoS performance of the current service user on other Web services to predict the QoS performance of other Web services, while IMEAN employs the average QoS performance of the Web service observed by other service users to predict the QoS performance for the current active user. UPCC only employs similar users for the QoS performance prediction by employing (7), while IPCC only employs similar Web services for the prediction by employing (8).

Table 2 shows the MAE and RMSE results of different prediction methods on *response time* and *failure rate* employing 10, 20, and 30 percent densities of the training matrix, respectively. For the active users, we vary the number of provided QoS values (*given number*) as 10, 20, and 30 by randomly removing entries (named as G10, G20, and G30,

TABLE 2
MAE and RMSE Comparison with Basic Approaches (a Smaller MAE or RMSE Value Means a Better Performance)

Metric	Density	Methods	Training Users = 100						Training Users = 140					
			Response Time			Failure Rate			Response Time			Failure Rate		
			G10	G20	G30	G10	G20	G30	G10	G20	G30	G10	G20	G30
MAE	10%	UMEAN	1623	1539	1513	5.71%	5.58%	5.53%	1521	1439	1399	5.01%	5.00%	4.97%
		IMEAN	903	901	907	2.40%	2.36%	2.46%	861	872	855	1.62%	1.58%	1.68%
		UPCC	1148	877	810	4.85%	4.20%	3.86%	968	782	684	4.11%	3.47%	3.28%
		IPCC	768	736	736	2.24%	2.16%	2.21%	585	596	605	1.39%	1.33%	1.42%
		WSRec	**758**	**700**	**672**	**2.21%**	**2.08%**	**2.08%**	**560**	**533**	**500**	**1.36%**	**1.26%**	**1.24%**
	20%	UMEAN	1585	1548	1508	5.74%	5.53%	5.51%	1464	1410	1390	5.21%	4.98%	4.95%
		IMEAN	866	859	861	2.36%	2.34%	2.29%	833	837	840	1.56%	1.61%	1.62%
		UPCC	904	722	626	4.40%	3.43%	2.85%	794	626	540	3.93%	2.96%	2.43%
		IPCC	606	610	639	2.01%	1.98%	1.98%	479	509	538	1.17%	1.22%	1.28%
		WSRec	**586**	**551**	**546**	**1.93%**	**1.80%**	**1.70%**	**445**	**428**	**416**	**1.10%**	**1.08%**	**1.07%**
	30%	UMEAN	1603	1543	1508	5.64%	5.58%	5.56%	1494	1430	1387	5.12%	4.98%	4.93%
		IMEAN	856	854	853	2.26%	2.29%	2.30%	823	823	827	1.56%	1.58%	1.58%
		UPCC	915	671	572	4.25%	3.25%	2.58%	803	576	491	3.76%	2.86%	2.06%
		IPCC	563	566	602	1.84%	1.83%	1.86%	439	467	507	1.10%	1.12%	1.17%
		WSRec	**538**	**504**	**499**	**1.78%**	**1.69%**	**1.63%**	**405**	**385**	**378**	**1.05%**	**1.00%**	**0.98%**
RMSE	10%	UMEAN	3339	3250	3192	15.47%	15.04%	14.74%	3190	3109	3069	14.75%	14.42%	13.99%
		IMEAN	1441	1436	1442	5.61%	5.58%	5.85%	1112	1140	1107	3.27%	3.26%	3.38%
		UPCC	2036	1455	1335	10.84%	7.51%	6.55%	1585	1174	1005	8.86%	5.42%	4.96%
		IPCC	1335	1288	1278	5.36%	5.27%	5.53%	850	871	867	2.87%	2.82%	2.96%
		WSRec	**1329**	**1247**	**1197**	**5.31%**	**5.12%**	**5.11%**	**819**	**789**	**734**	**2.80%**	**2.61%**	**2.61%**
	20%	UMEAN	3332	3240	3211	15.49%	15.05%	14.80%	3190	3124	3062	14.72%	14.24%	14.07%
		IMEAN	1269	1252	1257	4.67%	4.62%	4.54%	997	1001	1002	2.53%	2.61%	2.63%
		UPCC	1356	1128	1019	8.07%	5.31%	4.58%	1028	837	730	7.35%	4.20%	3.24%
		IPCC	1020	1016	1056	4.15%	4.13%	4.12%	664	700	731	2.00%	2.09%	2.19%
		WSRec	**997**	**946**	**937**	**4.04%**	**3.83%**	**3.67%**	**620**	**598**	**581**	**1.88%**	**1.84%**	**1.83%**
	30%	UMEAN	3336	3246	3197	15.49%	15.00%	14.68%	3178	3103	3086	14.68%	14.25%	14.07%
		IMEAN	1207	1209	1203	4.21%	4.23%	4.22%	955	954	957	2.28%	2.29%	2.28%
		UPCC	1267	1035	924	7.72%	5.09%	4.15%	988	741	644	6.49%	3.90%	2.66%
		IPCC	950	957	995	3.72%	3.71%	3.75%	611	642	685	1.73%	1.74%	1.81%
		WSRec	**921**	**884**	**869**	**3.64%**	**3.46%**	**3.37%**	**564**	**540**	**528**	**1.64%**	**1.55%**	**1.52%**

TABLE 3
MAE Comparison on Response Time (Smaller Value Means Better Performance)

Num. of Training Users	40			80			120		
Given Number	**10**	**20**	**30**	**10**	**20**	**30**	**10**	**20**	**30**
PQP	1424.87	1137.93	1011.99	1247.03	910.41	785.66	1090.58	882.01	801.31
SF	1113.10	1033.08	954.88	876.58	753.64	687.39	780.00	665.94	645.97
WSRec	944.72	896.14	858.22	752.46	688.54	632.08	666.36	627.11	601.30

TABLE 4
MAE Comparison on Failure Rate (Smaller Value Means Better Performance)

Num. of Training Users	40			80			120		
Given Number	**10**	**20**	**30**	**10**	**20**	**30**	**10**	**20**	**30**
PQP	4.69%	3.47%	2.89%	4.00%	2.95%	2.14%	4.34%	3.00%	2.17%
SF	3.26%	2.73%	2.49%	2.39%	2.06%	1.82%	2.32%	1.96%	1.73%
WSRec	3.11%	2.70%	2.47%	2.26%	2.02%	1.77%	2.12%	1.76%	1.60%

respectively, in Table 2). We also vary the number of training users as 100 and 140. We set $\lambda = 0.1$, since the item-based approach achieves better prediction accuracy than the user-based approach in our Web service QoS data set. The detailed investigation of the λ value setting will be shown in Section 5.8. Each experiment is run for 50 times and the average MAE and RMSE values are reported. We did not report the confidence interval of the experiments since those values are very small.

The experimental results of Table 2 shows that:

- Under all experimental settings, our *WSRec* method obtains smaller MAE and RMSE values consistently, which indicates better prediction accuracy.
- The MAE and RMSE values of *WSRec* become smaller with the increase of the given number from 10 to 30, indicating that the prediction accuracy can be improved by providing more QoS values.
- With the increase of the training user number from 100 to 140, and with the increase of the training matrix density from 10 to 30 percent, the prediction accuracy also achieve significant enhancement, since larger and denser training matrix provides more information for the prediction.
- The item-based approaches (IMEAN, IPCC) outperform the user-based approaches (UMEAN, UPCC). This observation indicates that similar Web services provide more information than similar users for the prediction in our user-item matrix.

In order to compare the Web service QoS value prediction performance of our approach with other state-of-the-art memory-based collaborative filtering approaches, extensive experiments are conducted. We compare with the following algorithms: Personalized QoS Prediction (PQP) [25], and Similarity Fusion (SF) [30]. PQP [25] predicts missing values by employing QoS data from similar users, while SF [30] predicts missing values by fusing the predictions from three sources, i.e., predictions based on ratings of the same item by other users, predictions based on different item ratings made by the same user, and predictions based on data from similar users on similar items. Tables 3 and 4 summarize our experimental results. As shown in these tables, our method outperforms the competitive approaches in various experimental settings.

5.4 Impact of the Missing Value Prediction

The *missing value prediction* in Section 4.2 makes use of the similar users and similar items to predict the missing values of the training matrix to make it more denser. Our *WSRec* method alleviates the potential negative influences of bad prediction on the missing data by not predicting the missing value if it has neither similar users nor similar items. To study the impact of the *missing value prediction*, we implement two versions of *WSRec*. One version employs missing value prediction while the other version does not. In the experiments, we vary the *given number* of the active users from 5 to 50 with a step value of 5 and vary the values of *training users* from 20 to 140 with a step value of 20. In reality, the training matrix is usually very sparse, therefore, we set the $density = 10\ percent$ to make the training matrix sparser. We also set *Top-K = 10*, which means that the top 10 similar neighbors will be employed for value prediction.

Fig. 4 shows the experimental results, where Figs. 4a, 4b, 4c, and 4d show the experimental results of *response time* and Figs. 4e 4f, 4g, and 4h show the experimental results of *failure rate*. Fig. 4 indicates that:

- *WSRec with missing value prediction* outperforms *WSRec without missing value prediction* consistently in all experimental settings, indicating that by predicting missing values for the training matrix, we are able to obtain more accurate prediction results.
- The prediction accuracies of both the two versions of *WSRec* enhance with the increase of given number and training user number. Since more QoS values and a larger training matrix provide more information for the missing value prediction.
- The same as the results shown in Table 2, the results of RMSE is following the same trend of MAE. Due to space limitation, in the following experiments, we only report the experimental results of MAE.

5.5 Impact of the Significance Weight

Significance weight makes the similarity computation more reasonable in practice by devaluing the similarities which look similar but are actually not. To study the impact of the *significance weight*, we implement two versions of WSRec, one version employs significance weight for the similarity computation, while the other version does not. In the

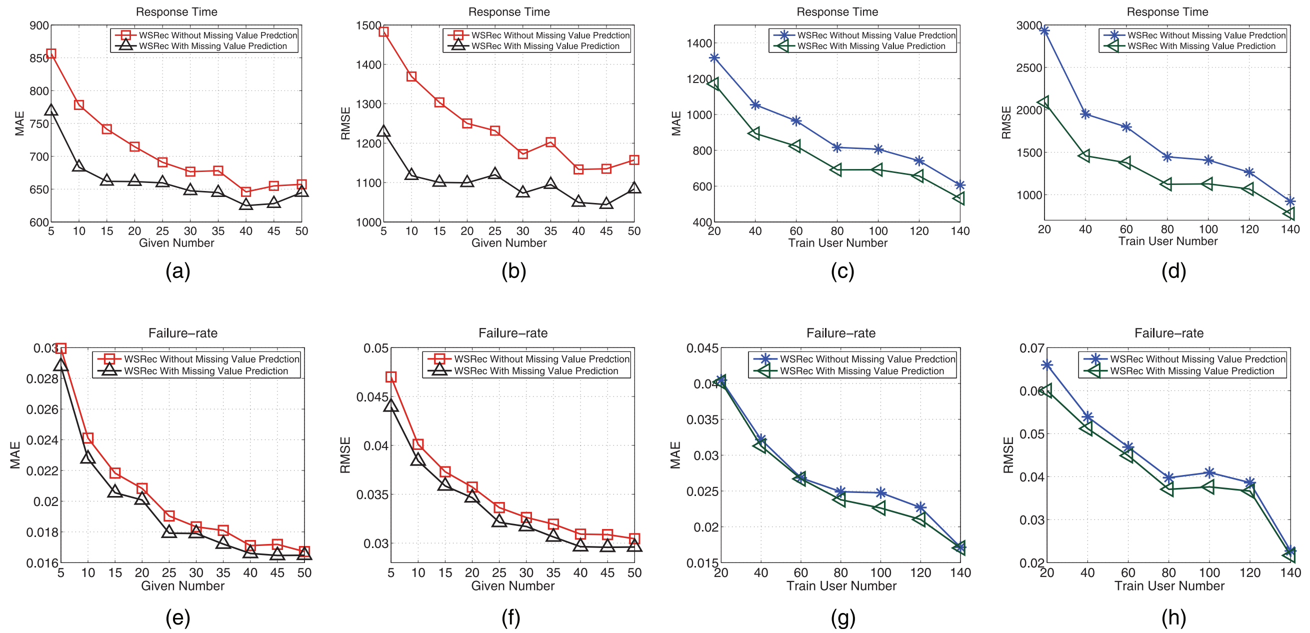

Fig. 4. Impact of the training matrix prediction.

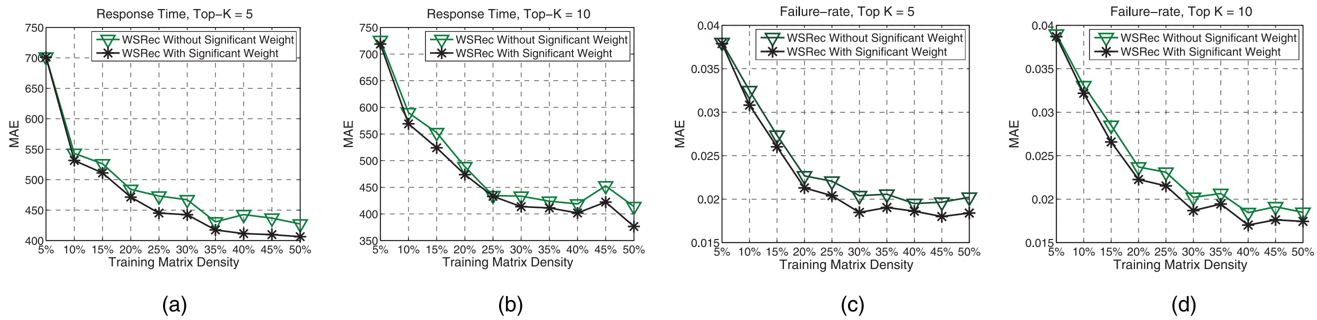

Fig. 5. Impact of the significance weight.

experiment, we set $given\ number = 5$, $\lambda = 0.1$, and $training\ users = 140$. We vary the density of the training matrix from 5 to 50 percent with a step value of 5 percent. We do not study the density value of 0, since in that case the training matrix contains no information and cannot be employed for the QoS value prediction.

Figs. 5a and 5c employ $Top\text{-}K = 5$, while Figs. 5b and 5d employ $Top\text{-}K = 10$. Fig. 5 shows that *WSRec with significance weight* obtains better prediction accuracy consistently than *WSRec without significance weight*. The improvement is not significant since the improvement of excluding dissimilar neighbors is alleviated by a lot of normal cases. The cases of excluding dissimilar neighbors do not happen very often comparing with the normal cases in our experiments.

As shown in Fig. 5, when the training matrix density increase, the prediction improvement of employing *significance weight* becomes more significant. Since with denser training matrix, more similar users will be found for the current user and the influence of excluding dissimilar users is thus becoming more significant.

5.6 Impact of the Confidence Weight

Confidence weight determines how to make use of the predicted values from the user-based method and the item-based method to achieve higher prediction accuracy automatically. To study the impact of the *confidence weight*, we also implement two versions of WSRec, one version employs *confidence weight*, while the other version does not. In the experiments, $Top\text{-}K = 10$ and $training\ users = 140$. We also set $\lambda = 0.5$, so that how to combine the user-based results and item-based results is not influenced by λ and is determined by the confidence weight alone.

Figs. 6a and 6c show the experimental results with given number change, while Figs. 6b and 6d show the experimental results with training matrix density change. As shown in Fig. 6, *WSRec with confidence weight* outperforms *WSRec without confidence weight* for both the *response time* and *failure rate*. Fig. 6 also shows that the MAE values become smaller with the increase of the given number and the training matrix density, which is consistent with the observation from Table 2.

5.7 Impact of Enhanced Top K

In our *WSRec* prediction method, we exclude dissimilar users with negative PCC values from the Top-K similar neighbors by using an enhanced Top-K algorithm. To study the impact of our enhanced Top-K algorithm on the prediction results, we implement two versions of *WSRec*. One version employs enhanced Top-K, while the other does not. Figs. 7a and 7c show the experimental results of response time and failure rate with given number change

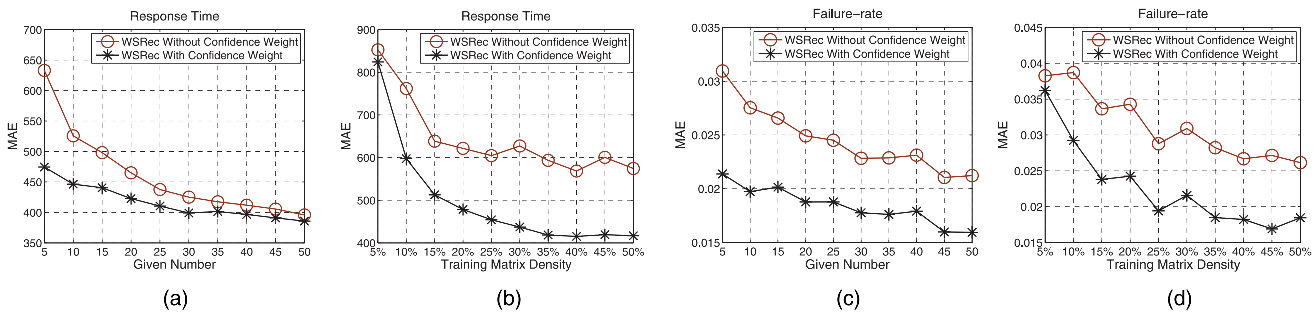

Fig. 6. Impact of the confidence weight.

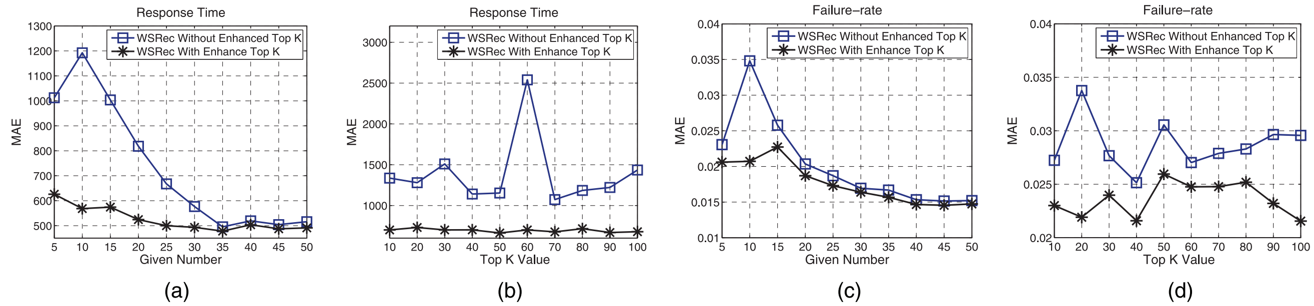

Fig. 7. Impact of the enhanced Top K.

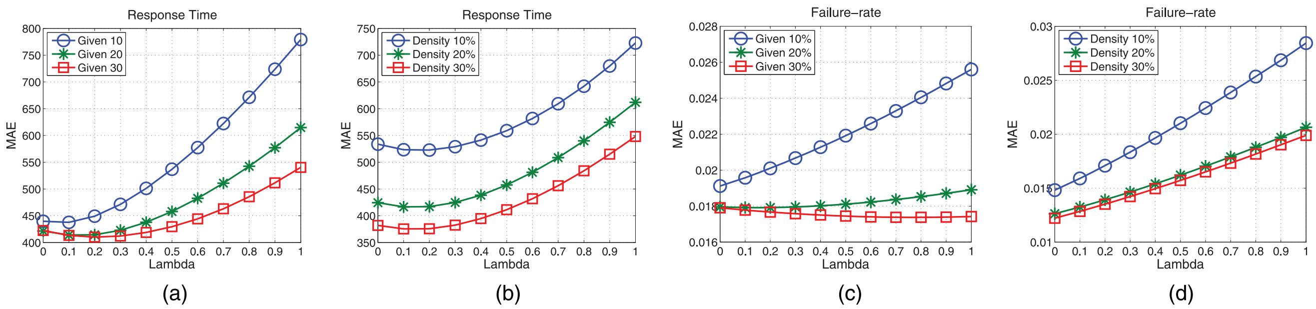

Fig. 8. Impact of the lambda.

under the experimental settings of $density = 10\,percent$, $training\ users = 140$, $\lambda = 0.1$, and $Top\text{-}K = 10$. Figs. 7b and 7d show the MAE values with top k value change under the experimental settings of $density = 10\,percent$, $given\ number = 5$, and $training\ users = 140$.

Fig. 7 shows that *WSRec with the enhanced Top-K* outperforms *WSRec without the enhanced Top-K* for both the *response time* and *failure rate*. The prediction performance of *WSRec without the enhanced Top-K* is not stable, since it may include dissimilar neighbors, which will greatly influence the prediction accuracy. Moreover, as shown in Figs. 7a and 7c, while the given number increases, differences of the two WSRec versions in MAE decrease. Since with larger given number, more similar users can be found for the current active user, the probability of selecting dissimilar users with negative PCC values as the top 10 similar user ($Top\text{-}K = 10$ in the experiment) is small. Our enhanced Top-K algorithm works only at situations that the number of similar users is smaller than the value of Top-K. Fig. 7 shows that the parameter Top-K can be set to be a large value for obtaining optimal performance in our WSRec approach.

5.8 Impact of λ

Different data sets may have different data correlation characteristics. Parameter λ makes our prediction method more feasible and adaptable to different data sets. If $\lambda = 1$, we only extract information from the similar users, and if $\lambda = 0$, we only consider valuable information from the similar items. In other cases, we fuse information from both similar users and similar items based on the value of λ to predict the missing value for active users.

To study the impact of the parameter λ to our collaborative filtering method, we set $Top\text{-}K = 10$ and $training\ users = 140$. We vary the value of λ from 0 to 1 with a step value of 0.1. Figs. 8a and 8c show the results of $given\ number = 10, 20$, and *30* with 20 percent density training matrix of *response time* and *failure rate*, respectively. Figs. 8b and 8d show the results of $density = 10, 20$, and $30\,percent$ with $given\ number = 20$ of *response time* and *failure rate*, respectively.

Observing from Fig. 8, we draw the conclusion that the value of λ impacts the recommendation results significantly, and a suitable λ value will provide better prediction accuracy. Another interesting observation is that, in Fig. 8a, with the given number increasing from 10 to 30, the optimal value of λ, which obtains the minimal MAE values of the curves in the figure, shifts from 0.1 to 0.3. This indicates that the optimal λ value is influenced by the given number. Similar to the observation in Figs. 8a and 8c, the optimal value of λ for *failure rate* shifts from 0 to 0.7, indicating that the optimal λ value is influenced not only by the given number, but also by the nature of data sets. For both the *response time* and *failure rate*, the similar items are more important than the similar users when limited Web service QoS values are given by the active users, while the similar users become more important when more QoS values are available from the active users. This observation is also confirmed by the experimental results reported in Table 2, where the IPCC outperforms the UPCC for all the $given\ number = 10, 20$, and 30. This is reasonable, since with limited user-given QoS values, the UMEAN prediction method, which employs the mean of the user-given QoS values to predict the QoS values of other Web services for this user, exhibits higher probability to be inaccurate. This will influence the prediction performance of UPCC, which is based on the value predicted by UMEAN for the missing value prediction as shown in (7).

As shown in Figs. 8b and 8d, with the given number of 20, all the three curves (*Density 10, 20, and 30 percent*) of *response time* and *failure rate* obtain the best prediction performance with the same λ value ($\lambda = 0.2$ for *response time* and $\lambda = 0$ for *failure rate*), indicating that the optimal λ value is not influenced by the training matrix density.

6 Related Work and Discussion

For presenting the nonfunctional characteristics of the Web services, QoS models of Web services have been discussed in a number of research investigations [13], [20], [21], [23], [29]. Based on the QoS performance of Web services, various approaches have been proposed for Web service selection [1], [5], [9], [33], [36], which enables optimal Web service to be identified from a set of functionally similar or equivalent Web service candidates. To obtain the values of the user-dependent QoS properties for a certain user, Web service evaluations from the client side are usually required [7], [18], [31]. To avoid the expensive real-world Web service invocations, our work employs the information of other similar service users as well as similar Web services to predict the QoS values for the active users.

Collaborative filtering methods are widely adopted in recommender systems [3], [17], [22]. Two types of collaborative filtering approaches are widely studied: memory based and model based. The most analyzed examples of memory-based collaborative filtering include user-based approaches [2], [10], [14], item-based approaches [8], [16], [24], and their fusion [30], [40]. User-based approaches predict the ratings of active users based on the ratings of their similar users, and item-based approaches predict the ratings of active users based on the computed information of items similar to those chosen by the active users. User-based and item-based approaches often use the PCC algorithm [22] and the VSS algorithm [2] as the similarity computation methods. PCC-based collaborative filtering generally can achieve higher performance than VSS, since it considers the differences in the user rating style. Wang et al. [30] combined user-based and item-based collaborative filtering approaches for movie recommendation. Different from Wang's work which uses similarity fusion, our approach considers the prediction confidence weights and design a parameter λ to determine how much our QoS value prediction approach relies on the user-based method and the item-based method. Moreover, our approach predicts the missing values for the training matrix first before the QoS value prediction for active users.

In the model-based collaborative filtering approaches, training data sets are used to train a predefined model. Examples of model-based approaches include the clustering model [32], aspect models [11], [12], [26] and the latent factor model [4]. Our collaborative filtering approach focuses on the memory-based methods since they are more intuitive to interpret the Web service recommendations. More investigations on the model-based approaches and imputation techniques for Web service QoS value prediction will be conducted in our future work. Different from the previous work [17], [22], [24] which mainly focuses on movie recommendation, our work provides a comprehensive study of how to provide accurate QoS value prediction for Web services.

There is limited work in the literature employing collaborative filtering methods for Web service QoS value prediction. One of the most important reasons that obstruct the research is that there is no large-scale real-world Web service QoS data sets available for studying the prediction accuracy. Without convincing and sufficient real-world Web service QoS data, the characteristics of Web service QoS information cannot be fully mined and the performance of the proposed algorithms cannot be justified. Work [15], [27] mention the idea of applying collaborative filtering methods to Web service recommendation and employs the MovieLens data set for experimental studies. However, employing the movie rating data set for studying Web service QoS value prediction is not convincing enough. Shao et al. [25] propose a user-based personalized QoS value prediction for Web services. In Section 5.3, we have shown that our approach outperforms this approach under different experimental settings.

Real-world Web service evaluations from distributed locations is not an easy task. In our previous work [36], [37], a real-world Web service evaluation has been conducted by five service users on eight publicly accessible Web services. Since the scale of this experiment is too small, the experimental results are not scalable for future research. In this paper, we conduct a large-scale real-world evaluation by involving 150 service users and 100 real-world Web services. 1.5 millions Web service invocation results are collected. This is the largest scale of QoS data that have even been collected for Web services. Our Web service QoS data set is released to promote future research and make our experimental study reproducible. This Web service QoS data set not only be employed for investigating Web service Qos value prediction, but also be employed for a lot of other QoS driven research topics, such as service selection [33], optimal service composition [34], fault tolerant Web services [38], composite service reliability prediction [39], Web service recommendation [40], and so on.

7 Conclusion

In this paper, we propose an approach for predicting QoS values of Web services by systematically combining the user-based PCC approach and the item-based PCC approach. Large-scale real-world experiments are conducted and the comprehensive experimental results show the effectiveness and feasibility of our approach.

Our ongoing research includes collecting QoS performance of more real-world Web services from more service users. More investigations will be conducted for QoS value updates, since the QoS values of Web services are changing from time to time in reality. In our Web service evaluations reported in this paper, to reduce the effect of the Web service invocations to the real-world Web services, we only selected one operation from a Web service for making evaluations and employ the performance of this operation to present the performance of the Web service. More investigations will be conducted on different operations of the same Web service 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 grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4154/09E and CUHK4128/08E).

References

[1] P.A. Bonatti and P. Festa, "On Optimal Service Selection," *Proc. 14th Int'l Conf. World Wide Web (WWW '04),* pp. 530-538, 2005.

[2] J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Ann. Conf. Uncertainty in Artificial Intelligence (UAI '98),* pp. 43-52, 1998.

[3] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction,* vol. 12, no. 4, pp. 331-370, 2002.

[4] J. Canny, "Collaborative Filtering with Privacy via Factor Analysis," *Proc. 25th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '02),* pp. 238-245, 2002.

[5] V. Cardellini, E. Casalicchio, V. Grassi, and F.L. Presti, "Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes," *Proc. Fifth Int'l Conf. Web Services (ICWS '07),* pp. 743-750, 2007.

[6] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM SIGCOMM Computer Comm. Rev.,* vol. 33, no. 3, pp. 3-12, July 2003.

[7] V. Deora, J. Shao, W. Gray, and N. Fiddian, "A Quality of Service Management Framework Based on User Expectations," *Proc. First Int'l Conf. Service-Oriented Computing (ICSOC '03),* pp. 104-114, 2003.

[8] M. Deshpande and G. Karypis, "Item-Based Top-N Recommendation," *ACM Trans. Information System,* vol. 22, no. 1, pp. 143-177, 2004.

[9] J.E. Haddad, M. Manouvrier, G. Ramirez, and M. Rukoz, "Qos-Driven Selection of Web Services for Transactional Composition," *Proc. Sixth Int'l Conf. Web Services (ICWS '08),* pp. 653-660, 2008.

[10] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering," *Proc. 22nd Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '99),* pp. 230-237, 1999.

[11] T. Hofmann, "Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis," *Proc. 26th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '03),* pp. 259-266, 2003.

[12] T. Hofmann, "Latent Semantic Models for Collaborative Filtering," *ACM Trans. Information System,* vol. 22, no. 1, pp. 89-115, 2004.

[13] M.C. Jaeger, G. Rojec-Goldmann, and G. Muhl, "Qos Aggregation for Web Service Composition Using Workflow Patterns," *Proc. Eighth IEEE Int'l Enterprise Computing Conf.,* pp. 149-159, 2004.

[14] R. Jin, J.Y. Chai, and L. Si, "An Automatic Weighting Scheme for Collaborative Filtering," *Proc. 27th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '04),* pp. 337-344, 2004.

[15] K. Karta, "An Investigation on Personalized Collaborative Filtering for Web Service Selection," Honours Programme thesis, Univ. of Western Australia, Brisbane, 2005.

[16] G. Linden, B. Smith, and J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," *IEEE Internet Computing,* vol. 7, no. 1, pp. 76-80, Jan./Feb. 2003.

[17] H. Ma, I. King, and M.R. Lyu, "Effective Missing Data Prediction for Collaborative Filtering," *Proc. 30th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07),* pp. 39-46, 2007.

[18] E. Maximilien and M. Singh, "Conceptual Model of Web Service Reputation," *ACM SIGMOD Record,* vol. 31, no. 4, pp. 36-41, 2002.

[19] M.R. McLaughlin and J.L. Herlocker, "A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience," *Proc. 27th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '04),* pp. 329-336, 2004.

[20] D.A. Menasce, "QoS Issues in Web Services," *IEEE Internet Computing,* vol. 6, no. 6, pp. 72-75, Nov./Dec. 2002.

[21] M. Ouzzani and A. Bouguettaya, "Efficient Access to Web Services," *IEEE Internet Computing,* vol. 8, no. 2, pp. 34-44, Mar./Apr. 2004.

[22] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," *Proc. ACM Conf. Computer Supported Cooperative Work,* pp. 175-186, 1994.

[23] S. Rosario, A. Benveniste, S. Haar, and C. Jard, "Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations," *IEEE Trans. Services Computing,* vol. 1, no. 4, pp. 187-200, Oct./Dec. 2008.

[24] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. 10th Int'l Conf. World Wide Web (WWW '01),* pp. 285-295, 2001.

[25] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS Prediction for Web Services via Collaborative Filtering," *Proc. Fifth Int'l Conf. Web Services (ICWS '07),* pp. 439-446, 2007.

[26] L. Si and R. Jin, "Flexible Mixture Model for Collaborative Filtering," *Proc. 20th Int'l Conf. Machine Learning (ICML '03),* pp. 704-711, 2003.

[27] R.M. Sreenath and M.P. Singh, "Agent-Based Service Selection," *J. Web Semantics,* vol. 1, no. 3, pp. 261-279, 2003.

[28] X. Su, T.M. Khoshgoftaar, X. Zhu, and R. Greiner, "Imputation-Boosted Collaborative Filtering Using Machine Learning Classifiers," *Proc. ACM Symp. Applied Computing (SAC '08),* pp. 949-950, 2008.

[29] N. Thio and S. Karunasekera, "Automatic Measurement of a Qos Metric for Web Service Recommendation," *Proc. Australian Software Eng. Conf.,* pp. 202-211, 2005.

[30] J. Wang, A.P. de Vries, and M.J. Reinders, "Unifying User-Based and Item-Based Collaborative Filtering Approaches by Similarity Fusion," *Proc. 29th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '06),* pp. 501-508, 2006.

[31] G. Wu, J. Wei, X. Qiao, and L. Li, "A Bayesian Network Based Qos Assessment Model for Web Services," *Proc. IEEE Int'l Conf. Services Computing (SCC '07),* pp. 498-505, 2007.

[32] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen, "Scalable Collaborative Filtering Using Cluster-Based Smoothing," *Proc. 28th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '05),* pp. 114-121, 2005.

[33] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient Algorithms for Web Services Selection with End-to-End Qos Constraints," *ACM Trans. Web,* vol. 1, no. 1, pp. 1-26, 2007.

[34] L. Zeng, B. Benatallah, A.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.,* vol. 30, no. 5, pp. 311-327, May 2004.

[35] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing.* Springer and Tsinghua Univ., 2007.

[36] Z. Zheng and M.R. Lyu, "A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services," *Proc. Sixth Int'l Conf. Web Services (ICWS '08),* pp. 145-152, 2008.

[37] Z. Zheng and M.R. Lyu, "Ws-Dream: A Distributed Reliability Assessment Mechanism for Web Services," *Proc. 38th Int'l Conf. Dependable Systems and Networks (DSN '08),* pp. 392-397, 2008.

[38] Z. Zheng and M.R. Lyu, "A Qos-Aware Fault Tolerant Middleware for Dependable Service Composition," *Proc. 39th Int'l Conf. Dependable Systems and Networks (DSN '09),* pp. 239-248, 2009.

[39] Z. Zheng and M.R. Lyu, "Collaborative Reliability Prediction for Service-Oriented Systems," *Proc. IEEE/ACM 32nd Int'l Conf. Software Eng. (ICSE '10),* 2010.

[40] Z. Zheng, H. Ma, M.R. Lyu, and I. King, "Wsrec: A Collaborative Filtering Based Web Service Recommender System," *Proc. Seventh Int'l Conf. Web Services (ICWS '09)*, pp. 437-444, 2009.

Zibin Zheng received the BEng and MPhil degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2005 and 2007, respectively. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He is serving as a program committee member of the first IEEE International Conference on Cloud Computing. He has also served as reviewer for international journals and conferences, e.g., the *IEEE Transactions on Software Engineering*, the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Services Computing*, WWW, DSN, ISSRE, SCC, etc. His research interests include service computing, software/service reliability engineering, and web technology. He is a student member of the IEEE.

Hao Ma received the BEng and MEng degrees from the School of Information Science and Engineering, Central South University, in 2002 and 2005, respectively. He worked as a system engineer at Intel Shanghai before joining the Chinese University of Hong Kong as a PhD student in Nov. 2006. He is currently working toward the PhD degree in the Computer Science and Engineering Department, Chinese University of Hong Kong. His research interests are in information retrieval, data mining, machine learning, social network analysis, and recommender systems.

Michael R. Lyu received the BS degree in electrical engineering from National Taiwan University, Taipei, ROC, in 1981; the MS degree in computer engineering from the University of California, Santa Barbara, in 1985; and the PhD 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. He is also the director of the Video over Internet and Wireless (VIEW) Technologies Laboratory. He was with the Jet Propulsion Laboratory as a technical staff member from 1988 to 1990. From 1990 to 1992, he was with the Department of Electrical and Computer Engineering, University of Iowa, Iowa City, as an assistant professor. From 1992 to 1995, he was a member of the technical staff in the applied research area of Bell Communications Research (Bellcore), Morristown, New Jersey. From 1995 to 1997, he was a research member of the technical staff at Bell Laboratories, Murray Hill, New Jersey. 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 more than 270 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* (Wiley, 1995) and *The Handbook of Software Reliability Engineering* (IEEE and New McGraw-Hill, 1996). He received Best Paper Awards at ISSRE 1998 and ISSRE 2003. He initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the program chair for ISSRE 1996, the general chair for ISSRE 2001, the program cochair for PRDC 1999, WWW 2010, SRDS 2005, and ICEBE 2007, the general cochair for PRDC 2005, and a program committee member for many other conferences including HASE, ICECCS, ISIT, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WI. He is frequently invited as a keynote or tutorial speaker to conferences and workshops in the US, Europe, and Asia. He has been on the editorial boards of the *IEEE Transactions on Knowledge and Data Engineering*, the *IEEE Transactions on Reliability*, the *Journal of Information Science and Engineering*, and the *Software Testing, Verification & Reliability Journal*. He is a fellow of the IEEE, AAAS, and Croucher Senior Research.

Irwin King received the BSc degree in engineering and applied science from the California Institute of Technology, Pasadena, in 1984. He received the MSc and PhD degrees 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. His research interests include machine learning, information retrieval, web intelligence and social computing, and multimedia processing. In these research areas, he has published more than 140 refereed journal (*JMLR, ACM TOIS, IEEE TNN, IEEE BME, PR, IEEE SMC, JAMC, JASIST, IJPRAI, NN,* etc.) and conference (NIPS, CIKM, SIGIR, IJCAI, ICML, IJCNN, ICONIP, ICDAR, WWW, etc.) manuscripts. In addition, he has contributed more than 20 book chapters and edited volumes and has more than 30 research and applied grants. One notable system he developed is the CUPIDE (Chinese University Plagiarism IDentification Engine) system, which detects similar sentences and performs readability analysis of text-based documents in both English and Chinese to promote academic integrity and honesty. He is an associate editor of the *IEEE Transactions on Neural Networks (TNN)*. He is a member of the editorial boards of the *Open Information Systems Journal, Journal of Nonlinear Analysis and Applied Mathematics*, and *Neural Information Processing Letters and Reviews Journal (NIP-LR)*. He also served as a special issue guest editor for *Neurocomputing*, the *International Journal of Intelligent Computing and Cybernetics (IJICC)*, the *Journal of Intelligent Information Systems (JIIS)*, and the *International Journal of Computational Intelligent Research (IJCIR)*. Currently, he is serving on the Neural Network Technical Committee (NNTC) and the Data Mining Technical Committee under the IEEE Computational Intelligence Society (formerly the IEEE Neural Network Society). He is also a vice-president and governing board member of the Asian Pacific Neural Network Assembly (APNNA). He is serving or has served as a program and/or organizing member in numerous top international conferences and workshops, e.g., WWW, ACM MM, CIKM, ICME, ICASSP, IJCNN, ICONIP, ICPR, etc. He has also served as reviewer for international conferences as well as journals, e.g., *Information Fusion*, *IEEE TCAS*, SIGMOD, the *IEEE Transactions on Neural Networks*, the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *IEEE Transactions on Multimedia*, the *IEEE Transactions on Knowledge and Data Engineering*, the *IEEE Transactions on System, Man, and Cybernetics*, *Machine Vision and Applications*, the *International Journal of Computer Vision, Real-Time Imaging*, the *SPIE Journal of Electronic Imaging*, the *International Journal of Pattern Recognition and Artifcial Intelligence*, etc. He is a member of the ACM, the IEEE Computer Society, the International Neural Network Society (INNS), and the Asian Pacific Neural Network Assembly (APNNA). He is a senior member of the IEEE.