A New QoS-Aware Web Service Recommendation System Based on Contextual Feature Recognition at Server-Side

Shun Li, Junhao Wen, Fengji Luo, *Member, IEEE*, Min Gao, Jun Zeng, and Zhao Yang Dong, *Fellow, IEEE*

***Abstract*—Quality of service (QoS) has been playing an increasingly important role in today's Web service environment. Many techniques have been proposed to recommend personalized Web services to customers. However, existing methods only utilize the QoS information at the client-side and neglect the contextual characteristics of the service. Based on the fact that the quality of Web service is affected by its context feature, this paper proposes a new QoS-aware Web service recommendation system, which considers the contextual feature similarities of different services. The proposed system first extracts the contextual properties from WSDL files to cluster Web services based on their feature similarities, and then utilizes an improved matrix factorization method to recommend services to users. The proposed framework is validated on a real-world dataset consisting of over 1.5 million Web service invocation records from 5825 Web services and 339 users. The experimental results prove the efficiency and accuracy of the proposed method.**

***Index Terms*—Web service, matrix factorization, Web service clustering, QoS prediction, recommendation system.**

I. Introduction

WEB SERVICES are self-contained Web applications, which contain standard interfaces to support interpretable, machine-to-machine interactions on the Internet [1]. Due to the explosion of available Web services on today's Internet, users are getting increasingly confused about choosing appropriate Web services. Therefore, it is highly desirable to develop techniques to automatically and efficiently make service selections and recommend them to users.

Manuscript received March 24, 2016; revised November 10, 2016, February 19, 2017, and March 3, 2017; accepted April 2, 2017. Date of publication April 13, 2017; date of current version June 9, 2017. The work was partially supported in part by the National Natural Science Foundation of China (Project No. 61379158), in part by the National Science and Technology Support Program (Project No. 2014BAH25F01), Chongqing Basic Research Program (Project No. cstc2014jcyjA40054), in part by the National Science Fund for Young Scholars (Project No. 61502062), and in part by the Early Career Development Scheme of Faculty of Engineering and Information Technology, University of Sydney, Australia. The associate editor coordinating the review of this paper and approving it for publication was M. Steinder.
(Corresponding author: Junhao Wen.)

S. Li, J. Wen, M. Gao, and J. Zeng are with the School of Software Engineering, Chongqing University, Chongqing 400044, China, and also with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing 400044, China (e-mail: lishun@cqu.edu.cn; jhwen@cqu.edu.cn; gaomin@cqu.edu.cn; zengjun@cqu.edu.cn).

F. Luo is with the School of Civil Engineering, University of Sydney, Sydney, NSW 2006, Australia (e-mail: fengji.luo@sydney.edu.au).

Z. Y. Dong is with the School of Electrical Engineering and Telecommunications, University of NSW, Sydney, NSW 2052, Australia (e-mail: zydong@ieee.org).

Digital Object Identifier 10.1109/TNSM.2017.2693324

Quality of service (QoS) has been widely used to qualify the non-functional characteristics of Web services and measure the users' satisfaction [2]–[5]. Across different Web services, QoS values can vary greatly. This is due to variance in Internet conditions as well as the heterogeneous physical environments of the users who use the services. As a result, many researchers have investigated the impacts of QoS and proposed many QoS-aware recommendation methods [6]–[8]. These methods have been proven to be capable of producing accurate prediction results with most of the methods based on the hypothesis that the QoS values of Web services invoked by an active user can be predicted based on the QoS data from other users who have similar service invocation records.Existing techniques [9], [10] can help users select the most suitable services that provide good non-functional performance. These methods only utilize the matrix of invocation qualities for each user and service pair to predict the quality of a Web service. This is partially because the invocation results are not complete QoS values without explicit information of functional features of Web services. However, in real applications, other associated contextual features of users and services (e.g., the categories of the functionality that a service provides) affect the quality of Web service as well [7]. Unlike existing works, this paper proposes a new QoS-aware Web service recommendation system which considers not only QoS metrics, but also functional categories of Web services. The proposed system can also reduce the impact of the cold-start problem [11], [12] by employing a matrix factorization approach that integrates the functional categories of the Web service and can yield better predictions than previous approaches [13], [14].

In summary, the major contributions of this paper are threefold:

- Proposal of a QoS-aware Web service recommendation system. At the client side, the proposed system takes into account the similarities of QoS values; at the server side, it discovers the functional patterns of Web services by analyzing the contextual characteristics from the WSDL files;
- Proposal of an improved prediction method for more accurate Web service recommendation. The proposed method is based on the matrix factorization technique, and considers both the historical Web service

invocation records of the users and contextual features of services;
- Presentation of results from extensive experiments conducted on real-word Web service QoS dataset consisting of more than 1.5 million Web service invocation records by 339 users from more than 20 countries.

The remainder of this paper is organized as follows. Section II introduces the mechanisms of the proposed Web service recommendation system; Section III discusses the experimental studies; Section IV reviews related works; finally, Section V concludes the paper and discusses the future works.

II. PRINCIPLES OF THE PROPOSED WEB SERVICE RECOMMENDATION SYSTEM

In this section, we firstly introduce our contextual feature extraction method for the Web service. Following this we present our approach to cluster Web services based on WSDL files, as well as the service recommendation method. Finally, the complexity analysis of the proposed method is presented and discussed.

A. QoS Recommendation Framework

QoS is an important factor to quantify the user's satisfaction on Web services. According to the work of Zheng *et al.* [6], the client-side QoS value is a critical factor to make accurate QoS value predictions for the active user based on the past Web service usage experiences. Therefore, in our recommendation system the QoS value is taken into account in the user similarity calculation, which is performed after the Web service contextual feature extraction and service clustering.

The basic idea of the proposed approach is depicted by Fig. 1. The features of Web services are firstly extracted from WSDL files with Web services then clustered based on their feature-level similarities. Next, users are clustered based on their QoS value similarities before being integrated with Web services into a Matrix Factorization model to allow QoS prediction.

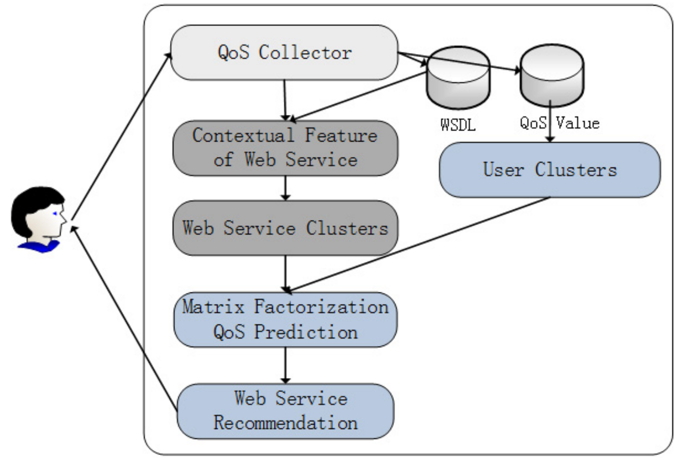

Fig. 1. QoS prediction framework.

B. Phase 1: Extract Web Service Contextual Feature

Contextual features describe the functionality category that a service belongs to. Following the work of Elgazzar *et al.* [15], we extract five most representative contextual features from WSDL files, i.e., WSDL content, WSDL type, WSDL message, WSDL port, and Web service name. These contextual features describe and reveal the functionality of a Web service, and play a significant role in the QoS recommendation.

1) Feature 1 (Content): WSDL files describe the functions of Web services. In our proposed approach, we process the content of WSDL documents and extract the vector of representative content words. The detailed processing procedures are as follows:

1. **Extract the practical vector:** The content of a WSDL file is firstly split into a vector based on the white spaces, and a practical vector is obtained by removing XML tags.
2. **Obtain stems:** In this step, all the words in the practical vector will be replaced by their base stem. This is based on the observation that words with a common stem usually have a similar meaning. In order to gather semantic word statistics, Porter stemmer [16] is employed to reduce all words to their base stem. Finally, a content vector is generated based on the basic words.
3. **Strip the function word:** There are both function words and content words in the vector, but only the content words are significantly relevant to the functions of Web services. In this step, the function words are removed. A general way to decide whether a word in the content vector is a function word or not is to compute the overestimation factor for each word based on the Poisson distribution [17]. The overestimation factor can be calculated as:

$$\Lambda_w = \frac{\overline{TF_w}}{TF} \tag{1}$$

where TF_w is the observed document frequency of the word w; $\overline{TF_w}$ is the number of estimated documents that contain the word w. Generally, the word with higher Λ_w is more likely a content word. In this paper, we define an overestimation factor threshold Λ^T as follows:

$$\Lambda^T = \begin{cases} avg(\Lambda) & if\ avg(\Lambda) > 1 \\ 1 & otherwise \end{cases} \tag{2}$$

The word with overestimation factor above Λ^T is considered as a content word. By removing the function words, a vector containing only content words is formed.
4. **Filter feature words:** Words with very high occurrence frequency are likely to be too general to distinguish different Web services. To get real feature words, some general words need to be removed from the content vector (e.g., 'data', 'port', etc.). In this paper we employ the TF-IDF approach to remove general words from the content vector of all WSDL files [18], [19]. Firstly, the inverse document frequency of a word is calculated as:

$$IDF_w = \log \frac{|D|}{\sum_{d \in D} I(W \in d)} \tag{3}$$

```
<wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://www.ankorus.com/webservices/numberling/">
    <s:element name="RepresentNumberTextural">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="number" type="s:unsignedLong" />
          <s:element minOccurs="0" maxOccurs="1" name="culture" type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>

<wsdl:message name="RepresentNumberTexturalSoapIn">
  <wsdl:part name="parameters" element="tns:RepresentNumberTextural" />
</wsdl:message>
<wsdl:message name="RepresentNumberTexturalSoapOut">
  <wsdl:part name="parameters" element="tns:RepresentNumberTexturalResponse" />
</wsdl:message>

<wsdl:portType name="MultiLingualNumberRepresentationSoap">
  <wsdl:operation name="RepresentNumberTextural">
    <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Returns textural representation
    of a number  (first parameter) in a culture (second parameter)</documentation>
    <wsdl:input message="tns:RepresentNumberTexturalSoapIn" />
    <wsdl:output message="tns:RepresentNumberTexturalSoapOut" />
  </wsdl:operation>
</wsdl:portType>

<wsdl:service name="MultiLingualNumberRepresentation">
```

Type

Message

Port

Service Name

Fig. 2. QoS Structure of WSDL Document.

where D is the number of WSDL files; $I(W \in d) = 1$ if $W \in d$; IDF_w represents the inverse document frequency of the word w. Then, the importance of the word w in the whole WSDL file can be calculated as:

$$TFIDF_w = TF_w * IDF_w \tag{4}$$

Following this, a threshold is set to decide whether a word should be considered as a general word or not. If so, then the word needs to be removed from the vector.

Finally, the final content vector can be obtained by applying the 4 steps sequentially. In this paper, the Normalized Google Distance (NGD) [20] is employed to compute the similarity of two words based on the word co-existence in the Web pages. NGD measures the semantic similarity between two words, and is known to outperform most other methods [20]. By applying NGD, the 'distance' between two words is calculated as:

$$dis(w, i) = 1 - NGD(w, i) \tag{5}$$

where w and i are the content words of service s_w and s_i respectively.

Then, the similarity of Web service contents is calculated as:

$$simCon(s_1, s_2) = \frac{\sum_{w \in CV_{s1}} \sum_{i \in CV_{s2}} dis(w, i)}{|CV_{s1}| \times |CV_{s2}|} \tag{6}$$

where CV_{s1} is the content vector of Web service s_1; $|CV_{s1}|$ is the cardinality of content vector CV_{s1}.

2) Feature 2 (Type): WSDL files contain two sections, name and type attribute of the service. Compared to the type attribute, the name attribute is not always effective at showing functional features. For a WSDL file, its type attribute is often organized as a complex structure with an array of elements (e.g., the 'RepresentNumberTextural' type in Fig. 2). Therefore, we extract the elemental type values and compute the similarity between a pair of Web services based on (7):

$$simType(s_1, s_2) = \frac{2 \times Match(s_1, s_2)}{|TV_{s1}| + |TV_{s2}|} \tag{7}$$

where TV_{s1} is the set of defined types in the WSDL file s_1; $|TV_{s1}|$ is the length of TV_{s1}; $MatchTy(s_1, s_2)$ denotes the function which calculates the number of matched type values between the two services s_1 and s_2.

3) Feature 3 (Message): The message segment of a WSDL file consists of one or more parameters that are passed between different Web services. The parameters in a message are associated with a service type, as illustrated in Fig. 2. The message definition (including the name and type of the message) is typically a refined abstraction of the content from a message. Fig. 2 gives two message definition examples: '*RepresentNumberTexturalSoapIn*' and 'RepresentNumberTexturalSoapOut'. Similar with (7), we compute the similarity between two Web services in the message level.

4) Feature 4 (Port): The portType element defines the detailed content and sequence of the message for an operation. Fig. 2 shows an example of portType '*MultiLingualNumberRepresentationSoap*' with an operation *RepresentNumberTextural*. The attributes of the portType are the message defined in the message section. Therefore, similar with the computations of type-level and message-level similarities, the similarity of Web services in the aspect of portType can be also calculated by (7).

5) Feature 5 (Service Name): The service name element often represents the functions of a Web service, and as such is considered an important feature. Generally, the service name consists of some feature words. We firstly split these feature words into multiple word segments. For example, the service name 'MultiLingualNumberRepresentation' in Fig. 2 can be separated into four words: Multi, Lingual, Number, and Representation. Then, we compute the similarity between service names by using (5) and (6).

C. Phase 2: Web Service Clustering

In this section, we use the FCM (Fuzzy C-Means) [21] algorithm to cluster Web services based on the service feature vectors presented above. At the end of each clustering iteration, each Web service is assigned a membership value for each cluster center. Then, the similarity between Web services can be measured as:

$$\begin{aligned} simS(s_i, s_j) = \lambda_1 simCon(s_i, s_j) + \lambda_2 simType(s_i, s_j) \\ \lambda_3 simMsg(s_i, s_j) + \lambda_4 simPort(s_i, s_j) \\ + \lambda_5 simName(s_i, s_j) \end{aligned} \tag{8}$$

where λ_1, λ_2, λ_3, λ_4, and λ_5 are weights of the five features of the Web service. In this study, we set $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 0.2$. After the clustering, a set of similar Web services $S(i)$ for a Web service s_i can be calculated as (9), where C(i) is the cluster for which the services in $S(i)$ have maximal membership values.

$$S(i) = \left\{ s \middle| s \in C(i), S_i \in C(i), s! = S_i \right\}. \tag{9}$$

D. Phase 3: User Clustering Based on Neighborhood Similarity

The similarity of users can be calculated from the QoS values provided by different users who invoke the same Web service. In some cases, QoS values of a certain user may be missing. The missing values can be predicted by employing other QoS values observed by similar users. By employing the Pearson Correlation Coefficient (PCC), the similarities between two users can be computed based on their observed QoS values on the commonly invoked Web services:

$$SimU(i,k) = \frac{\sum_{j\in I(i)\cap I(k)} \left(R_{ij} - \bar{R}_i\right) * \left(R_{kj} - \bar{R}_k\right)}{\sqrt{\sum_{j\in I(i)\cap I(k)} \left(R_{ij} - \bar{R}_i\right)^2} * \sqrt{\sum_{j\in I(i)\cap I(k)} \left(R_{kj} - \bar{R}_k\right)^2}} \tag{10}$$

where $I(i)$ is the set of Web services invoked by user U_i; R_{ij} denotes the QoS value of the Web service S_j observed by user U_i; $\overline{R_i}$ represents the average QoS value rate of Web services observed by user U_i. From (10), it can be seen that the user similarity $Sim(i,k)$ is within $[-1, 1]$, where a larger value indicates users U_i and U_k are more similar. We then use the function $f(x) = (x + 1)/2$ to map the PCC similarity into the range of [0, 1].

According to [27], the positive user neighborhood cluster can be obtained by using the Top-K strategy based on similar users to the current user. However, in our approach both positive and negative users are taken into account and are mapped into the range of [0, 1]. This mapping strategy is based on the consideration that the negative users would also affect the prediction for the current user. A set of similar users $U(i)$ (or the user neighborhood cluster) for a service user U_i can be described as:

$$U(i) = \{U_k | U_k \in Top - K(U_i), i! = k\} \tag{11}$$

where $Top-K(U_i)$ is a set of the most $Top-K$ similar users of the current service user. Based on the user neighborhood information and Web service information, the QoS value prediction can be performed.

E. Phase 4: Web Service Contextual Characteristics and Neighborhood Integrated Matrix Factorization

Matrix factorization is a widely adopted approach to predict missing values. It factorizes the user-item rating matrix into a low-rank matrix to undertake prediction [38]. This factorization is based on the premise that there would be a small number of latent factors influencing the user's preference on Web services. In addition, each latent factor can have a significant influence on the user's usage experience and preference on a Web service when it is applied on the user and Web service. In this paper, matrix factorization is utilized to predict the unknown QoS values and services with good quality attributes recommended to users.

In this paper, we define a rating matrix to represent the user's numerical ratings on items, and use a low-rank matrix factorization approach to approximate the rating matrix by a multiplication of rank factors. We consider a matrix factor $U \in IR^{l\times m}$ and $V \in IR^{l\times n}$ where $R \approx U^T V$. Each column of U denotes the factor vector for a specific user in the matrix with ℓ latent features representing the information of the specific user. Moreover, V shows the items matrix with ℓ latent features where each the column is a liner predictor for a service. Therefore, the missing QoS values can be estimated by linear combinations of the factor vectors of the user-specific matrix U and item-specific matrix V. Then by calculating the accumulated error between the actual QoS matrix and the inner product of U and V, we have the following object function:

$$\begin{aligned} \min_{w,v} l(R, U, V) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^{R} \left(R_{ij} - U_i^T V_j\right)^2 \\ + \frac{\lambda_u}{2} \|U\|_F^2 + \frac{\lambda_v}{2} \|V\|_F^2 \end{aligned} \tag{12}$$

where I_{ij}^R is the indicator function that is equal to 1 if the user U_i invokes the Web service V_j, and it is equal to 0 otherwise. $\|.\|_F^2$ denotes the Fresenius norm which penalizes large values of U and V; U and V are parameters which need to be estimated. To avoid over fitting, two regularization terms $\frac{\lambda_u}{2}\|u\|_F^2$ and $\frac{\lambda_v}{2}\|v\|_F^2$ are added into Model (12).

The above approach is the traditional collaborative filtering (CF) approach which uses all available QoS values to predict missing values. However, due to the sparsity of the real Web service QoS matrix, in most cases the traditional matrix factorization cannot generate optimal QoS values. Therefore, in this paper we utilize the performance information of similar Web services and user neighbors in the QoS value factorizing process in Phase 2 and Phase 3. A high performance approach is proposed in this study to minimize the sum of squared errors and quadratic regularization terms, which is formulated as:

$$\begin{aligned}\min_{u,s} l(R, U, S) = &\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}\left(R_{ij} - U_i^T S_j\right)^2 \\ &+ \frac{\alpha}{2}\sum_{i=1}^{m}\sum_{k\in U(i)} SimU(i,k)\|U_i - U_k\|_F^2 \\ &+ \frac{\beta}{2}\sum_{j=1}^{n}\sum_{q\in S(j)} SimS(j,q)\left\|S_j - S_q\right\|_F^2 \\ &+ \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_s}{2}\|S\|_F^2 \end{aligned} \tag{13}$$

where $\alpha, \beta > 0$. $U(i)$ is a set of similar users of the user U_i; $S(j)$ is the Web service performance category to which S_j belongs; $SimU(i, k)$ is the user similarity calculated by (10); $SimS(j, q)$is the Web service feature similarity calculated by (8). $SimS(j, q)$ is not calculated by the number of users who invoke the service because similarity of the QoS values cannot be ensured. Model (13) includes two regularization terms. The first one minimizes the tastes between user U_i and its similar users, shown as (14):

$$\frac{\alpha}{2}\sum_{i=1}^{m}\sum_{k\in U(i)} SimU(i,k)\|U_i - U_k\|_F^2 \tag{14}$$

The second term minimizes the performances between Web service S_j and its similar Web services, and can be expressed as:

$$\frac{\beta}{2}\sum_{j=1}^{n}\sum_{q\in S(j)} SimS(j,q)\left\|S_j - S_q\right\|_F^2 \tag{15}$$

The gradient descent approach is then applied on latent feature vectors U_i and S_j to get the local minimum of the following objective function:

$$\begin{aligned}\frac{\partial \ell}{\partial U_i} = &\sum_{j=1}^{n} I_{ij}^{R}\left(U_i^T S_j - R_{ij}\right)S_j + \lambda_U U_i \\ &+ \alpha\sum_{k\in U(i)} SimU(i,k)(U_i - U_k)\end{aligned} \tag{16}$$

$$\begin{aligned}\frac{\partial \ell}{\partial S_j} = &\sum_{i=1}^{m} I_{ij}^{R}\left(U_i^T S_j - R_{ij}\right)U_i + \lambda_S S_j \\ &+ \beta\sum_{q\in S(j)} SimS(j,q)\left(S_j - S_q\right).\end{aligned} \tag{17}$$

F. Complexity Analysis

The proposed algorithm mainly focuses on the QoS value prediction with m users and n services. In the prediction phase, the main computation of the gradient methods is to evaluate the object function ℓ and its gradients against the variables. We set R_τ to be the number of existing values in the matrix, K to be the number of similar neighbors, and $\hat{Q}$ to be the maximum number of similar services for the different sizes of Web service clusters, respectively. The computational complexity of evaluating the object function ℓ is therefore $O(R_\tau l + R_\tau K l + R_\tau \hat{Q} l)$, where l is the dimension of the latent feature between the user and service. The computational complexities of the gradients $\frac{\partial \ell}{\partial U}$ and $\frac{\partial \ell}{\partial S}$ in (16) and (17) are $O(R_\tau K l + R_\tau K^2 l)$ and $O(R_\tau \hat{Q}^2 l + R_\tau \hat{Q} l)$, respectively. Therefore, the total computational complexity in one iteration is determined by the maximal value of $O(R_\tau K l + R_\tau K^2 l)$ and $O(R_\tau \hat{Q}^2 l + R_\tau \hat{Q} l)$. The computational time of our approach has a first-order linear correlation with the existing observations in the matrix $\boldsymbol{R}$. This proves that our approach is efficient and can be well adapted for large datasets.

III. Experiments

In this section, we discuss extensive experiments conduct to validate the proposed QoS-aware Web service recommendation system.

A. Experimental Setup

We use a real-world Web service QoS dataset, WS-Dream, collected by Zheng *et al.* [22] for experimentation. This dataset contains over one and a half millions Web service invocation records from 339 users and 5,825 Web services with full WSDL files. We divide the 339 users into two groups, with 80% of users as training data and the other 20% as test data, respectively. To simulate the real situation, we randomly remove a certain number of RTT records from the training user group to obtain a sparse training matrix.

Two well-known statistical accuracy metrics, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), are used to evaluate the prediction accuracy of the proposed method. MAE is the average absolute deviation of predictions to the ground truth data. For all test services and test users, MAE is calculated as:

$$MAE = \frac{\sum_{i,j}\left|R_{ij} - \hat{R}_{ij}\right|}{N} \tag{18}$$

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 smaller value of MAE indicates better prediction result.

RMSE measures the differences between the actual and predicted values, expressed as:

$$RMSE = \sqrt{\frac{\sum_{i,j}\left|R_{ij} - \hat{R}_{ij}\right|^2}{N}}. \tag{19}$$

B. Experimental Comparison and Analysis

We compare our approach with 5 other state-of-the-art CF algorithms:

- UPCC: This well recognized approach involves using similar user behaviors to make predictions [23].
- IPCC: This method is widely used in e-commerce systems. In Web service recommendation systems it utilizes similar services to make predictions [24].
- UIPCC: This method combines UPCC and IPCC [25].

TABLE I
RESULT COMPARISON BETWEEN FAMOUS PREDICTION APPROACHES

METHODS	MATRIX DENSITY= 5%		MATRIX DENSITY= 10%		MATRIX DENSITY= 15%		MATRIX DENSITY= 20%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
UPCC	0.5989	1.4039	0.5509	1.2880	0.5205	1.2777	0.4917	1.2517
IPCC	0.6243	1.4342	0.5892	1.3149	0.5324	1.2960	0.4136	1.2043
UIPCC	0.5859	1.3970	0.5022	1.2767	0.5137	1.2525	0.4056	1.1685
PMF	0.6001	1.4779	0.5345	1.3167	0.4831	1.2726	0.4313	1.1047
NIMF	0.5606	1.3864	0.4889	1.2490	0.4575	1.2206	0.4303	1.1105
WSFNIMF	**0.5516**	**1.3851**	**0.4885**	**1.2226**	**0.4360**	**1.2128**	**0.4240**	**1.0422**

- PMF: This method applies the probabilistic matrix factorization method on the user-item matrix to generate recommendations [26].
- NIMF: As proposed by Zheng *et al.* [27] this state-of-the-art method uses matrix factorization.

In our experiments, to simulate real world situations, we randomly remove some entries from the original matrix. The densities of the matrix vary from 5% to 20% with the step of 5%. The 5% density means only 5% of the values are used as the training data and the other 95% are removed from the matrix to be the testing data. Then we compare our method with above five methods on the same training dataset. The parameters of our method are set as: $\alpha = 0.001$, $\beta = 0.001$, $\lambda_u = \lambda_s = 0.001$, Top-K = 10, and dimensionality = 50. The general comparison results are reported in Table I. The reason for these parameter settings will be explained in Sections III-C–III-G.

From Table I, it can be seen that the MAE and RMSE of our proposed method are significantly smaller than those of other methods. This indicates the higher prediction accuracy of the proposed method. Meanwhile, the MAE and RMSE of our method became slightly smaller with the increase of matrix density. Among all the prediction methods, our proposed method shows superior prediction performance than others. This proves the realization that by utilizing both the features in the server side and the neighborhood information in NIMF [27], better prediction performance can be achieved. Generally, the experimental results validate our assumption that different Web service features would affect the user's choices on the services, and that the missing QoS values can be exactly predicted by integrating the contextual information.

C. Impact of Parameter α

In our system, the parameter α controls the weight of the user relationship in the recommendation model. If the value of α is too large, the proposed method will mainly focus on the contributions of the neighborhood users with general MF model and will thus underestimate the importance of the performance of Web services. If the value of α is too small, the Web service feature information would dominate the prediction process, which would potentially decrease the prediction accuracy. In this section, we conduct a sensitivity study of α under conditions of dimensionality = 50, $\beta = 0.001$, with matrix densities varying from 5% to 20%. The results are shown in Fig. 3.

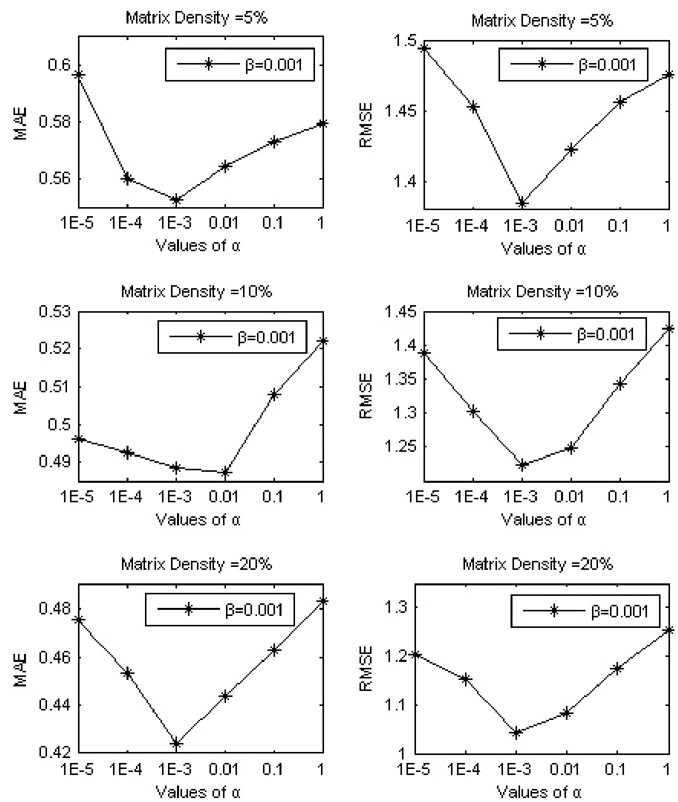

Fig. 3. Impact of Parameter α.

Fig. 3 shows that the MAE and RMSE values decrease at the beginning of the experiment until a certain threshold is reached, and then increase again with the increase of α. This observation proves that the better prediction performance can be achieved by integrating both the client side QoS information and service feature information into the matrix factorization, rather than only considering the information of one side.

From Fig. 3, it can be seen that the proposed method achieves best MAE and RMSE values when $\alpha = 0.001$. This finding indicates that better recommendation performance can be achieved by considering a combination of information from both server and client sides rather than solely utilizing one side information. In the following experiments, we set α to be 0.001.

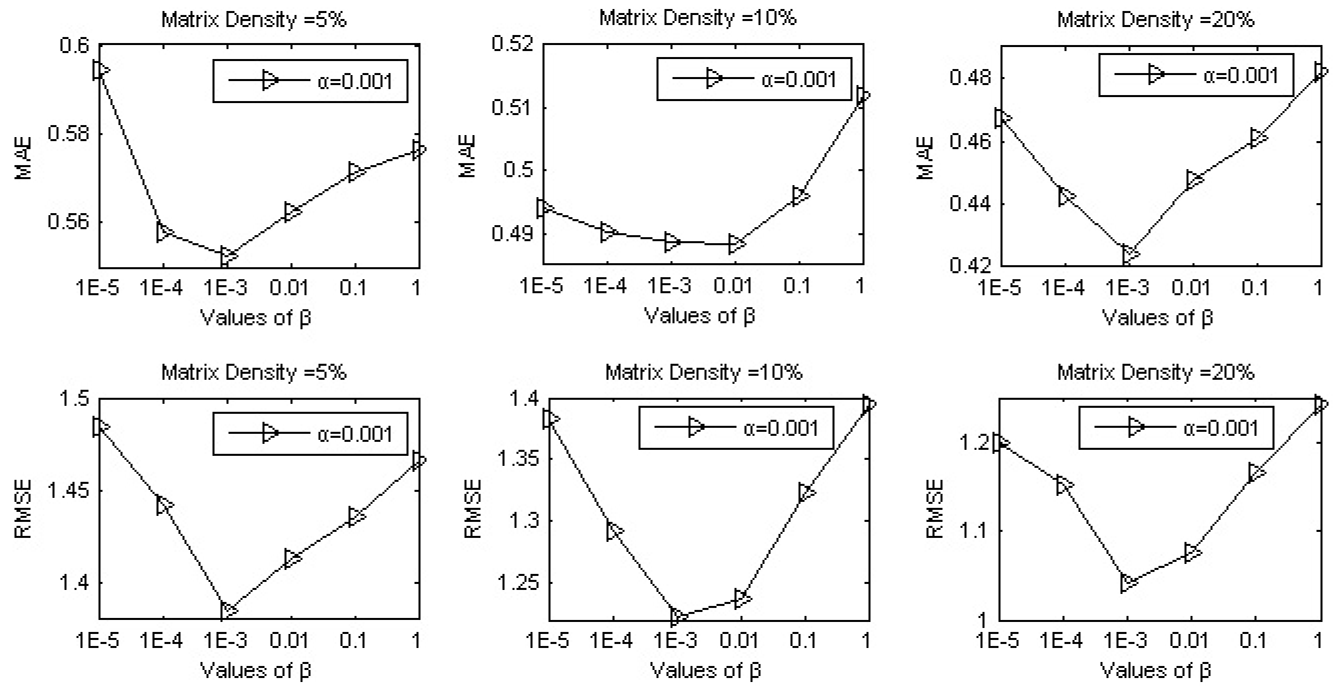

Fig. 4. Impact of Parameter β.

D. Impact of Parameter β

Parameter β controls the weight of the server side information in the prediction process. On the one hand, too large a value of β would force the prediction to heavily rely on the service features and neglect the importance of the user neighborhood information; on the other hand, if β is too small, the Web service feature information would make little contribution in the prediction process. We conduct the sensitivity study for β under conditions of $\alpha = 0.001$ (the best optimal value in Section III-C), dimensionality $= 50$, and matrix density varying from 5% to 20%. The results are reported in Fig. 4.

From Fig. 4, we can see that the MAE and RMSE values firstly decrease, and then increase with the increase of β. Meanwhile, the MAE value in the sensitivity study of β is smaller than that of α. This clearly proves our intuition that Web service feature information is useful when developing appropriate recommendation model.

Fig. 4 shows that the optimal setting of β is approximately 0.001, under which the best MAE and RMSE values are obtained. This indicates that the Web service feature can significantly affect the performance of our system.

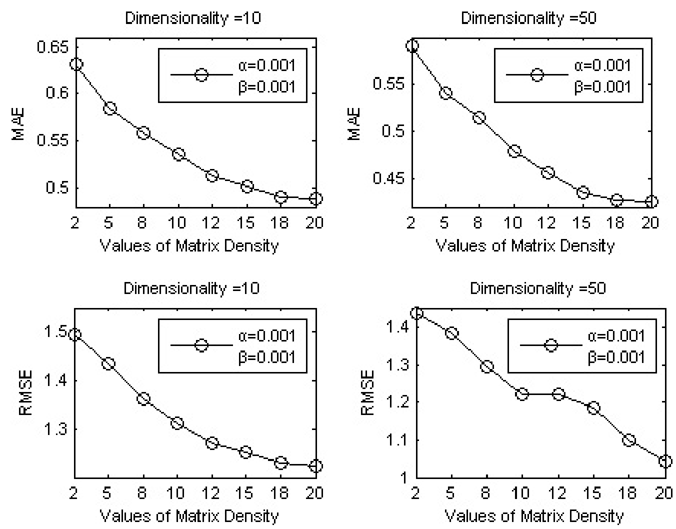

Fig. 5. Impact of Matrix Density.

E. Impact of Matrix Density

In this section, we perform the sensitivity study for the matrix density when it changes from 2% to 20%, under conditions of $\alpha = 0.001$, $\beta = 0.001$ (the best optimal value in Section III-D), and Top-K $= 10$. Two dimensionality settings are studied, which are 10 and 50, respectively. The results are shown in Fig. 5.

Fig. 5 shows that both the MAE and RMSE values decrease with the increase of the matrix density, indicating the increase of the prediction accuracy. In the beginning, both MAE and RMSE decrease sharply. Then, the rate of decrease slows with the increase of the matrix density. The sensitivity study results show when the training data is very sparse, if more information can be utilized (i.e., additional entries added into the matrix), better prediction accuracy can be achieved.

F. Impact of Dimensionality

The dimensionality parameter controls the number of latent factors utilized in the proposed system. We perform the sensitivity study of the dimensionality under conditions of $\alpha = 0.001$, $\beta = 0.001$, Top-K $= 10$, and matrix density varying from 5% to 20%. The results are reported in Fig. 6.

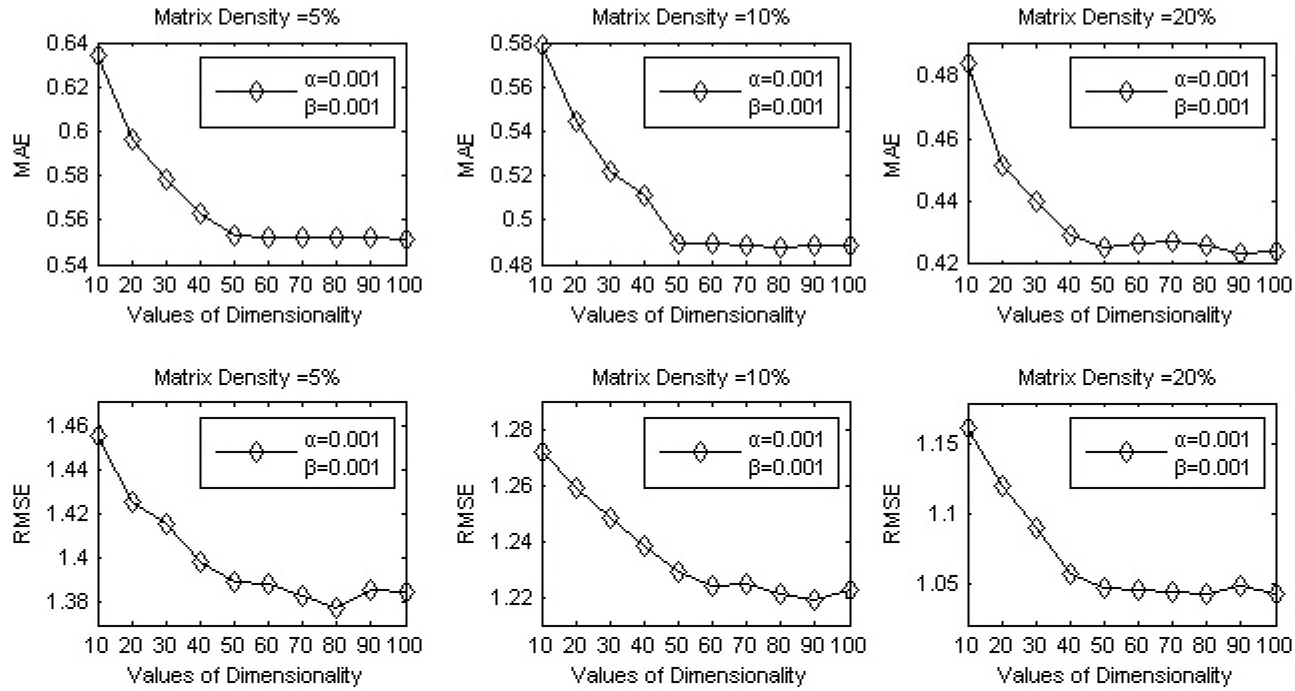

Fig. 6. Impact of Dimensionality.

Fig. 6 shows that the MAE and RMSE values decrease dramatically at the beginning, with the rate of the decrease then slowing. The results show that after a certain threshold, the increase of the dimensionality cannot always improve the prediction accuracy due to the increase of noise in the data introduced by the large dimensionality. In our test dataset, the threshold is approximately 50.

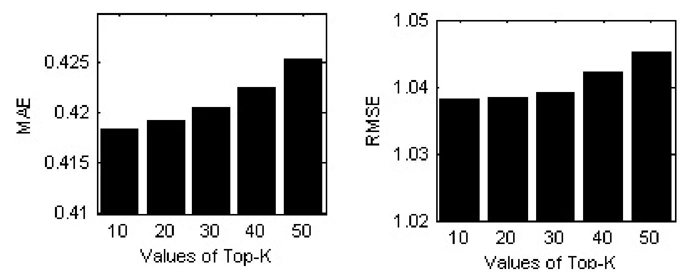

Fig. 7. Impact of Top-K.

G. Impact of Top-K

The Top-K value determines the number of similar neighbors employed in the proposed system. To study how the Top-K values affect the prediction results, we set the values of Top-K varying from 10 to 50. We perform the sensitivity study of the Top-K under conditions of $\alpha = 0.001$, $\beta = 0.001$, matrix density $= 20\%$, and dimensionality $= 50$.

Fig. 7 shows that the MAE and RMSE values slightly increase when the Top-K value increases from 10 to 50. The results show that large Top-K value will reduce the prediction accuracy due to the increase of noise in the data introduced by other dissimilar users. In both MAE and RMSE evaluations, it is clearly seen that the best results are achieved when Top-K = 10. Therefore, we set Top-K = 10 as the default parameter setting.

IV. Related Work and Discussion

A. Web Services Recommendation

Web service discovery and selection is an active topic in the service computing domain. Al-Masri and Mahmoud [39] proposed a novel method to rank Web services based on the user's preferences and QoS parameters. Kritikos and Plexousakis [40] proposed an effective QoS-based Web service discovery method based on the QoS defined in the context of Web services. Many recommendation techniques have also been proposed that focus on generating personalized Web service recommendation. For example, Deng *et al.* [48] computed similarity through the user's interaction, and proposed a Web service recommendation approach with trust enhancement in the social network. However, these methods were not validated using real world datasets.

Several Web service recommendation methods though have been test using real world Web service datasets. For example, Zheng *et al.* [6] proposed a novel approach called WSRec to predict the missing QoS values. Their method is based on collaborative filtering to create a Web service recommendation, and is validated using a real world Web service dataset released by the authors. Chen *et al.* [44] used location region to cluster users geographically, and computed the similarity of different location regions to generate a recommendation. Similarly, Jiang *et al.* [9] proposed an effective recommendation approach. In their method, the influences of the personalized services on user similarity computing are considered. They also considered the influences of different services when computing the service similarity.

However, when predicting the QoS value, all existing approaches only utilize the QoS values as the input data and neglect other contextual information. Building on these works, this paper retrieves the QoS information from both QoS values at the client-side and contextual features at the server-side, and in doing so achieves higher prediction accuracy.

B. Matrix Factorization

Collaborative filtering is a very popular recommendation approach in commercial recommendation systems, such as Amazon.com [28]–[31]. Generally, collaborative filtering can be divided into two categories: memory-based models and model-based models. The Memory-based models, such as KNN, perform recommendations based on computing the user similarity and item similarity on the entire user-item matrix [45]–[47]. These algorithms make full use of the entire user-item matrix to predict the best result for the active user. However, these methods often have high computational and memory costs, which limit their online applications. The model-based models, such as Neural Networks [32], [33], Bayesian models [34], [35], etc., firstly extract the learning model from the training dataset through machine learning techniques, and then apply the models for prediction and recommendation. These models perform better than memory-based models in creating online recommendations.

Matrix factorization is a widely adopted model-based approach in the recommender system study [36]–[38]. The Matrix factorization methods factorize the user-item matrix to find out two or more low-rank matrixes to make further predictions. Mnih and Salakhutdinov [26] presented a detailed probabilistic explanation for matrix factorization methods, and conducted experiments using a large-scale dataset from Netflix. Similarly, Zhang *et al.* [41] considered user information and added a user clustering regularization term to the standard matrix factorization to get the personalized items for users. Ma *et al.* [42] improved the matrix factorization model by integrating a social regularization term into the social network. Matrix factorization models have also been gradually appearing in the Web service recommendation domain. For example, Zheng *et al.* [27] analyzed the historical Web service usage information at the client-side and applied the user-collaboration on the Matrix Factorization as a constraint. Lo *et al.* [43] integrated the location-based regularization and geographical information between users to the matrix factorization. However, these models only utilized user information to improve the recommendation accuracy and ignore service features. In our recommendation approach, we consider not only user information, but also service features at the server side.

V. Conclusion

This paper proposes a new Web service recommendation system to make personalized QoS value predictions. The proposed approach takes full advantage of the user preference information and Web service features simultaneously to achieve higher prediction accuracy than other approaches. The proposed method firstly extracts service features from WSDL files at the server side, and then integrates QoS values at the client side to predict the unknown QoS value as shown in Figure 1. Extensive experimental analyses on a large real world dataset were conducted to validate the effectiveness of the proposed method.

The results of this paper suggest a number of promising avenues for future research. For example designing and creating recommendation systems that consider the impacts of Web service locations and infer a customers' personal preferences from social tags. The applications of the proposed framework within the mobile environment are also deserving of future investigation. In this paper, we use WSDL files to extract service features. This is due to the lack of contextual information on the real-world datasets. The development of online applications to collect more contextual information from Web services such as tags, time, and so forth is currently being explored.

Acknowledgment

The authors would like to acknowledge Dr. Kyle Holmes from the University of Newcastle, Australia for his valuable assistance on the proofreading and language checking of this paper.

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, *Web Services*. Heidelberg, Germany: Springer, 2004.

[2] A. M. Karande and D. R. Kalbande, "Web service selection based on QoS using tModel working on feed forward network," in *Proc. IEEE Int. Conf. Issues Challenges Intell. Comput. Tech. (ICICT)*, Ghaziabad, India, 2014, pp. 29–33.

[3] S. L. Liu, L. Yun-Xiang, Z. Fan, T. Gui-Fen, and J. Ning, "Dynamic Web services selection algorithm with QoS global optimal in Web services composition," *J. Softw.*, vol. 18, no. 3, pp. 646–656, 2007.

[4] D. Ardagna and B. Pernici, "Global and local QoS constraints guarantee in Web service selection," in *Proc. 3rd Int. Conf. Web Services (ICWS)*, Orlando, FL, USA, 2005, pp. 805–806.

[5] A. F. M. Huang, C.-W. Lan, and S. J. H. Yang, "An optimal QoS-based Web service selection scheme," *Inf. Sci.*, vol. 179, no. 19, pp. 3309–3322, 2009.

[6] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware Web service recommendation by collaborative filtering," *IEEE Trans. Services Comput.*, vol. 4, no. 2, pp. 140–152, Apr./Jun. 2011.

[7] R. Zhang, C. Li, H. Sun, Y. Wang, and J. Huai, "Quality of Web service prediction by collective matrix factorization," in *Proc. 11th Int. Conf. Services Comput. (SCC)*, Anchorage, AK, USA, 2014, pp. 432–439.

[8] C. Mao and J. Chen, "QoS prediction for Web services based on similarity-aware slope one collaborative filtering," *Informatica*, vol. 37, no. 2, pp. 139–148, 2013.

[9] Y. Jiang, J. Liu, M. Tang, and X. Liu, "An effective Web service recommendation method based on personalized collaborative filtering," in *Proc. 9th Int. Conf. Web Services (ICWS)*, Washington, DC, USA, 2011, pp. 211–218.

[10] K. O. Lee, S. Y. Kim, and K. C. Park, "QoS evaluation of KT-NGN," in *Proc. 9th Asia–Pac. Conf. Commun. (ICWS)*, vol. 3. Penang, Malaysia, 2003, pp. 900–903.

[11] Z.-K. Zhang, C. Liu, Y.-C. Zhang, and T. Zhou, "Solving the cold-start problem in recommender systems with social tags," *Europhys. Lett.*, vol. 92, no. 2, 2010, Art. no. 28002.

[12] T. Qiu, G. Chen, Z.-K. Zhang, and T. Zhou, "An item-oriented recommendation algorithm on cold-start problem," *Europhys. Lett.*, vol. 95, no. 5, 2011, Art. no. 58003.

[13] L. Liu, F. Lecue, N. Mehandjiev, and L. Xu, "Using context similarity for service recommendation," in *Proc. 4th Int. Conf. Semantic Comput. (ICSC)*, Pittsburgh, PA, USA, 2010, pp. 277–284.

[14] N. Thio and S. Karunasekera, "Automatic measurement of a QoS metric for Web service recommendation," in *Proc. 16th Aust. Conf. Softw. Eng. (ASWEC)*, Brisbane, QLD, Australia, 2005, pp. 202–211.

[15] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering WSDL documents to bootstrap the discovery of Web services," in *Proc. 8th Int. Conf. Web Services (ICWS)*, Miami, FL, USA, 2010, pp. 147–154.

[16] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[17] K. Church and W. Gale, "Inverse document frequency (IDF): A measure of deviations from Poisson," in *Proc. 3rd ACL Workshop Very Large Corpora*, 1995, pp. 121–130.

[18] C. C. Tapang. *Web Services Description Language (WSDL) Explained*, Microsoft Developer Netw., accessed on Apr. 15, 2017. [Online]. Available: http://www.fdi.ucm.es/profesor/jjruz/WebSI/Bibliografia/WSDL1.pdf

[19] E. Christensen. *Web Services Description Language (WSDL) 1.1*. Accessed on Apr. 15, 2017. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.467&rep=rep1&type=pdf

[20] R. L. Cilibrasi and P. M. B. Vitanyi, "The Google similarity distance," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 370–383, Mar. 2007.

[21] J. Bezdek, R. Ehrlich, and F. William, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, nos. 2–3, pp. 191–203, 1984.

[22] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world Web services," in *Proc. 17th Int. Conf. Web Services (ICWS)*, Miami, FL, USA, 2010, pp. 83–90

[23] K.-Y. Jung and J.-H. Lee, "Prediction of user preference in recommendation system using associative user clustering and Bayesian estimated value," in *Proc. Aust. Joint Conf. Artif. Intell.*, pp. 284–296, 2002.

[24] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, Hong Kong, 2001, pp. 285–295.

[25] M. Papagelis and D. Plexousakis, "Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents," *Eng. Appl. Artif. Intell.*, vol. 18, no. 7, pp. 781–789, 2005.

[26] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.

[27] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative Web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 289–299, Jul./Sep. 2013.

[28] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE. Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[29] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[30] H. Goto, Y. Hasegawa, and M. Tanaka, "Efficient scheduling focusing on the duality of MPL representation," in *Proc. IEEE Symp. Comput. Intell. Sched. (SCIS)*, Honolulu, HI, USA, Apr. 2007, pp. 57–64.

[31] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proc. 8th Int. ACM CSCW Conf. Comput. Supported Cooperative Work (CSCW)*, Philadelphia, PA, USA, 2000, pp. 241–250.

[32] J. Mai, Y. Fan, and Y. Shen, "A neural networks-based clustering collaborative filtering algorithm in e-commerce recommendation system," in *Proc. Int. Conf. Web Inf. Syst. Min. (WISM)*, Shanghai, China, 2009, pp. 616–619.

[33] G.-I. Jung, S.-S. Park, Y.-G. Shin, and D.-S. Jang, "Construction of personalized recommendation system based on back propagation neural network," *J. Korea Contents Assoc.*, vol. 7, no. 12, pp. 292–302, 2007.

[34] H.-S. Park, M.-H. Park, and S.-B. Cho, "Mobile information recommendation using multi-criteria decision making with Bayesian network," *Int. J. Inf. Technol. Decis. Making*, vol. 14, no. 2, pp. 317–338, 2015.

[35] Y. Ming and W. Deng, "A utility-based recommendation approach for E-commerce websites based on Bayesian networks," in *Proc. 2nd Int. Conf. Bus. Intell. Financ. Eng. (BIFE)*, Beijing, China, 2009, pp. 571–574.

[36] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[37] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proc. SIAM Int. Conf. Data Min. (SDM)*, Columbus, OH, USA, 2010, pp. 199–210.

[38] Y. Shi, M. Larson, and A. Hanjalic, "Mining mood-specific movie similarity with matrix factorization for context-aware recommendation," in *Proc. Workshop Context Aware Movie Recommendation ACM*, Barcelona, Spain, 2010, pp. 34–40.

[39] E. Al-Masri and Q. H. Mahmoud, "QoS-based discovery and ranking of Web services," in *Proc. 16th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Honolulu, HI, USA, 2007, pp. 529–534.

[40] K. Kritikos and D. Plexousakis, "Requirements for QoS-based Web service description and discovery," *IEEE Trans. Services Comput.*, vol. 2, no. 4, pp. 320–337, Oct./Dec. 2009.

[41] C.-X. Zhang *et al.*, "Information filtering via collaborative user clustering modeling," *Phys. A Stat. Mech. Appl.*, vol. 396, pp. 195–203, Feb. 2014.

[42] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proc. 4th Int. ACM Conf. Web Search Data Min. (WSDM)*, Honolulu, HI, USA, 2011, pp. 287–296.

[43] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative Web service QoS prediction with location-based regularization," in *Proc. 19th Int. Conf. Web Services (ICWS)*, Honolulu, HI, USA, 2012, pp. 464–471.

[44] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized QoS-aware Web service recommendation and visualization," *IEEE Trans. Services Comput.*, vol. 6, no. 1, pp. 35–47, Jan. 2013.

[45] G. Biau, B. Cadre, and L. Rouviere, "Statistical analysis of k-nearest neighbor collaborative recommendation," *Ann. Stat.*, vol. 38, no. 3, pp. 1568–1592, 2010.

[46] G. Lu, X. Ji, J. Li, and D. Yuan, "Difference factor' KNN collaborative filtering recommendation algorithm," in *Advanced Data Mining and Applications*. Cham, Switzerland: Springer, 2014, pp. 175–184.

[47] X. Luo, Y.-X. Ouyang, Z. Xiong, and M. Yuan, "The effect of similarity support in k-nearest-neighborhood based collaborative filtering," *Chin. J. Comput.*, vol. 33, no. 8, pp. 1437–1445, 2010.

[48] S. G. Deng, L. T. Huang, J. Wu, and Z.-H. Wu, "Trust-based personalized service recommendation: A network perspective," *J. Comput. Sci. Technol.*, vol. 29, no. 1, pp. 69–80, 2014.

Shun Li received the B.Eng. degree in 2014. He is currently pursuing the Ph.D. degree with the School of Software Engineering, Chongqing University. His research interests include service computing, recommendation system, and machine learning.

Junhao Wen received the Ph.D. degree from Chongqing University in 2008, where he is a Vice Head and a Professor with the School of Software Engineering. His research interests include service computing, cloud computing, and software dependable engineering. He has published over 80 refereed journal and conference papers in the above areas. He has over 30 research and industrial grants and developed many commercial systems and software tools.

Fengji Luo received the B.S. and M.S. degrees in software engineering from Chongqing University, China, in 2006 and 2009, respectively, and the Ph.D. degree in electrical engineering from the University of Newcastle, Australia, in 2013. He is currently a Post-Doctoral Research Associate with the School of Civil Engineering, University of Sydney, Australia. His research interests include computational intelligence, evolutionary computation, renewable energy, and smart grid applications. He has published over 20 IEEE TRANSACTIONS papers in the above areas. He was a recipient of one of the eight awardees of the "2016 Australia-Japan Emerging Research Leader Exchange Program."

Min Gao received the M.S. and Ph.D. degrees in computer science from Chongqing University in 2005 and 2010, respectively, where she is an Associate Professor with the School of Software Engineering. She was a Visiting Researcher with the School of Business, University of Reading. Her research interests include recommendation system, service computing, and data mining. She has published over 30 refereed journal and conference papers in the above areas. She has grants from the National Natural Science Foundation of China, the China Post-Doctoral Science Foundation, and the China Fundamental Research Funds for the Central Universities.

Jun Zeng received the B.S. and M.S. degrees in software engineering from Chongqing University in 2007 and 2010, respectively, and the Ph.D. degree in advanced information technology from Kyushu University in 2013. Since 2013, he has been a Lecturer with the School of Software Engineering, Chongqing University. His research interest includes Web data extraction, data mining, and search engine.

Zhao Yang Dong (M'99–SM'06–F'07) received the Ph.D. degree from the University of Sydney, Australia, in 1999. He is currently a Professor with the School of Electrical Engineering and Telecommunications, University of New South Wales, Australia. He was the Head of the School and a Professor with the University of Sydney, and the Ausgrid Chair and the Director of the Center for Intelligent Electricity Networks, University of Newcastle, Australia. He also held academic and industrial positions with Hong Kong Polytechnic University, Hong Kong, and Transcend Networks (currently, TASNetworks), Tasmania, Australia. His research interests include power system operation & planning, smart grid, renewable energy systems, computational intelligence, and data mining. He is an Editor of the IEEE TRANSACTIONS ON SMART GRID, the IEEE TRANSACTIONS ON SUSTAINABLE ENERGY, the IEEE PES LETTERS, and *IET Renewable Power Generation*.