Hindawi
Complexity
Volume 2020, Article ID 8572161, 17 pages
https://doi.org/10.1155/2020/8572161

Research Article

Quality Prediction of Web Services Based on a Covering Algorithm

Ying Jin,[1] Guangming Cui,[2] and Yiwen Zhang[3]

[1]*Department of Management, Hefei University, Hefei 230601, China*
[2]*School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia*
[3]*School of Computer Science and Technology, Anhui University, Hefei 230039, China*

Correspondence should be addressed to Yiwen Zhang; zywahu@qq.com

Received 2 December 2019; Revised 18 January 2020; Accepted 27 January 2020; Published 18 February 2020

Guest Editor: Yuan Yuan

Service-oriented architecture (SOA) is widely used, which has fueled the rapid growth of Web services and the deployment of tremendous Web services over the last decades. It becomes challenging but crucial to find the proper Web services because of the increasing amount of Web services. However, it proves unfeasible to inspect all the Web services to check their quality values since it will consume a lot of resources. Thus, developing effective and efficient approaches for predicting the quality values of Web services has become an important research issue. In this paper, we propose UIQPCA, a novel approach for hybrid User and Item-based Quality Prediction with Covering Algorithm. UIQPCA integrates information of both users and Web services on the basis of users' ideas on the quality of coinvoked Web services. After the integration, users and Web services which are similar to the target user and the target Web service are selected. Then, considering the result of integration, UIQPCA makes predictions on how a target user will appraise a target Web service. Broad experiments on WS-Dream, a web service dataset which is widely used in real world, are conducted to evaluate the reliability of UIQPCA. According to the results of experiment, UIQPCA is far better than former approaches, including item-based, user-based, hybrid, and cluster-based approaches.

1. Introduction

Service-oriented architecture (SOA) has become a significant tool when building intricate software systems by finding and clustering loosely connected Web services provided by different organizations [1, 2]. Executed by a system engine, e.g., a BPEL engine [3], the component services of such a service-oriented system (SOS) collectively realize the functionality of the system [4] which is often offered as SaaS (Software-as-a-Service) in the cloud environment [5, 6]. Under the help of cloud computing, business models such as e-business and e-commerce with pay-as-you-go model in particular are developing quickly and getting popular, which stimulates the quick development of Web services [7]. The statistics from ProgrammableWeb (http://www.programmableweb.com/), an online Web service directory, demonstrate that the number of published Web services has grown rapidly during the past few years. Because of the widespread use of Web services and SOA, we can build varied SOSs to meet the growing sophisticated business needs of various organizations [8, 9].

The quality of an SOS relies on its component services. To ensure the quality of an SOS, we must be able to predict the quality of the Web services used to compose the SOS, e.g., their response time, throughput, etc. Once Web services' quality can be predicted, Web service recommendation systems can be built and employed to recommend Web services with appropriate quality values that fulfil system engineers' quality requirements [10, 11]. Thus, quality prediction for Web services has been an extremely active research field in recent years [11–15]. Among many prediction approaches, collaborative filtering has been the most popular and successful one for building recommendation systems [16–18]. Memory-based collaborative filtering, which uses known preferences of some users to predict preferences of unknown users, has been widely employed by many researchers for the quality prediction on Web services [12, 13]. It predicts the quality of a given Web service for a target user, by using the historical quality values of Web

services which belong to other users. Here, the users of a Web service refer to the systems/applications that use the Web service.

In this research, given a user u, we define the users who have similar opinions as user u on a same set of Web services commonly invoked by themselves and user u as $S(u)$. Given a Web service i, we define the Web services which have left same impression on a same group of users similar to user i as$S(i)$. Accordingly, memory-based collaborative filtering approaches can be further categorized into the following three groups:

(1) User-based collaborative filtering: given a user u and a Web service i, user-based collaborative filtering techniques find $S(u)$ and use the experience on the quality of Web service i by $S(u)$ to predict how user u will like Web service i [19].

(2) Item-based collaborative filtering (in the context of this research, terms "item" and "services" are interchangeable): given a user u and a Web service i, item-based collaborative filtering techniques find $S(i)$ and use the quality of $S(i)$ to predict how user u will like Web service i [20, 21].

(3) Hybrid collaborative filtering: combining user-based and item-based techniques, hybrid collaborative filtering uses both $S(u)$ and $S(i)$ to predict how user u will like Web service i [14, 15].

The fundamental assumption of memory-based collaborative filtering (referred to as collaborative filtering in short hereafter) is that *if two users u andvrank n items similarly, or do similar things (e.g., buying, watching, listening, and consuming), they will have similar feedback on other items* [22, 23]. Thus, the key to accurate predictions based on collaborative filtering is to identify similar users and similar Web services [18, 24]. Several approaches have employed the k-means clustering method to cope with this problem by grouping similar users and similar Web services [25–27]. K-means is an unsupervised method, which parts given data points into a number (k) of clusters that can be chosen manually, with one randomly selected initial center for each cluster [28]. K-means-based quality prediction methods are mainly limited in two aspects. Firstly, in order to enable that the corresponding clustering results can show the similarity between users and Web services correctly, it is essential to solve the problem of finding an appropriate k, which can be very difficult. Secondly, the clustering results are closely related to initial centers randomly selected. Thus, the quality predictions of k-means-based methods are unreliable in both stability and accuracy.

Considering this problem, we propose UIQPCA, a novel hybrid User and Item-based Quality Prediction approach for Web services based on a Covering Algorithm without the requirement of a prespecified number of clusters or initial centers. Based on a set of users and Web services, UIQPCA firstly uses a covering algorithm to sort users into different groups according to their similarity in their opinions on the quality of coinvoked Web services. Similarly, UIQPCA then divides partition Web services into groups according to each user's appraisal of them. Next, to predict target user u's reflection on the quality of target Web service i, UIQPCA selects a large number of users and Web services based on the results of clustering which have the largest similarity to the user u and the Web service i. Finally, UIQPCA predicts how will user u appraise Web service.

In a word, the major contributions of this research are shown in the following aspects:

(1) UIQPCA employs an improved covering-based clustering algorithm to partition similar users and similar Web services into clusters with both effectiveness and efficiency. This clustering method can be executed without a prespecified or the manually selected initial centers. Thus, UIQPCA ensures stable predictions.

(2) Based on the clustering results, UIQPCA employs a unique way to quickly identify users and Web services which have the largest similarity to the target user and target Web service. Through this approach, we can determine the most appropriate k value by conducting experiments with different k values, which saves a lot of time by avoiding reclustering users and Web services.

(3) We have done a lot of experiments on the basis of the dataset of WS-Dream in order to make a comparison between UIQPCA and eight existing methods. The results demonstrate UIQPCA's significant advantages over those approaches in effectiveness and efficiency.

The remainder of this paper follows the following organization. Section 2 formally states the research topic, defines relevant concepts, and shows a motivating case. Section 3 discusses the technical details of UIQPCA. Section 4 experimentally compares UIQPCA with existing approaches about their predicting efficacy on the quality of Web services. Finally, Section 5 reviews the connected work and Section 6 summarizes this paper.

2. Issue Introduction and Motivating Example

Firstly, we give formal definition for two concepts, similar users and similar Web services.

Definition 1. Similar Users. Given a user u, we define the users who have similar opinions as user u on a same set of Web services commonly invoked by themselves and user u as $S(u)$.

Definition 2. Similar Web Services. Given a Web service i, we define the Web services which have left same impression on a same group of users similar to user i as $S(i)$.

Given a user u, a Web service i, a set of users $U = \{u_1, u_2, \ldots, u_m\}$, and a set of Web services $I = \{i_1, i_2, \ldots, i_n\}$, the predicting process of the quality of Web service i is mainly made up with two parts: (1) to identify $S(u)$ from U and $S(i)$ from I; (2) to predict user u's opinion on the quality of Web service i with reference to $S(u)$ and $S(i)$. This 2-phase procedure is depicted in Figure 1.

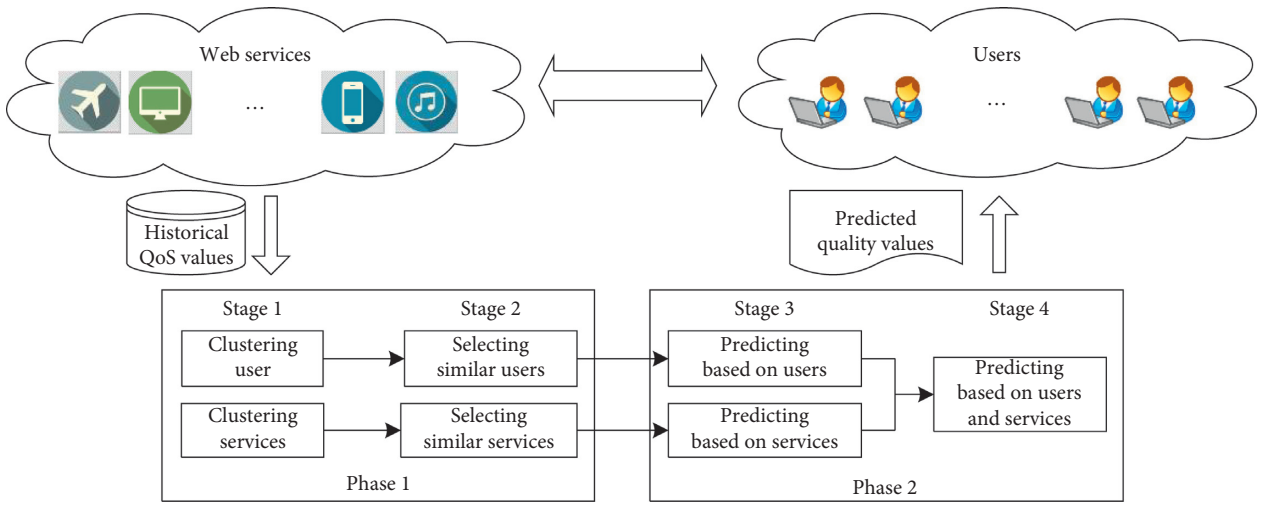

Figure 1: Quality prediction procedure.

Figure 2 shows the case of 9 users, $u_1, \ldots, u_9$, and 9 Web services, $i_1, \ldots, i_9$. The table includes response time the users have responded to the Web services, which is called a *user-item matrix*, represented as $M = [q_{u,i}]_{m \times n}$, in which m represents the amount of users and n represents the amount of Web services. In the matrix, each element $q_{u,I}$ is the one-dimensional quality value of Web service i when user u invocated i in the past, $1 \le u \le m$, $1 \le i \le n$. Here, $q_{u,i}$ can be the quality value obtained after one invocation or the average quality value obtained after multiple invocations. An empty cell $q_{u,i}$ shows that user u has not invocated web service i, so user u does not know its quality. The set of Web services that user u has invocated in the past is represented by $I(u)$, and the set of users that have invocated a Web service i is represented by $U(i)$. Take the user-item matrix in Figure 2 for example; there are $I(u_6) = \{i_1, i_3, i_4, i_5\}$ and $U(i_3) = \{u_1, u_2, u_5, u_6\}$.

For example, if we want to predict how user u_9 will appraise the quality of Web service i_4, we can represent the result as $q_{9,4}$. According to Figure 2, we can get the conclusion that $I(u_4) \cap I(u_9) = \{i_6, i_7, i_8, i_9\}$, which means that both users u_4 and u_9 have used Web services i_6, i_7, i_8, and i_9. If users u_4 and u_9's past quality experiences on i_6, i_7, i_8, and i_9 are similar, they are likely to have similar opinions on the quality of i_4. In this way, we can get $q_{4,4}$, which indicates how user u_4 feels about the quality of service i_4. Thus, taking into account the similarity between $\{q_{4,6}, q_{4,7}, q_{4,8}, q_{4,9}\}$ and $\{q_{9,6}, q_{9,7}, q_{9,8}, q_{9,9}\}$ into account, together with $q_{4,4}$, $q_{9,4}$ can be predicted. In the same way, $q_{4,2}$ can also be predicted similarly. If we look at users u_2 and u_9, there is $I(u_2) \cap I(u_9) = \varnothing$, meaning that the services invoked by users u_2 and u_9, respectively, do not overlap. Based on Definition 1, users u_2 and u_9 are not similar. Thus, u_2 is not helpful and should not be used in the prediction of $q_{9,4}$. If we want to improve the reliability of predicting $q_{9,4}$, we need to include as many users which are similar to u_9 as we can. Consequently, the identification of users similar to u_9, e.g., u_4, is of great importance. In the meantime, to exclude users dissimilar to u_9, e.g., u_2, is also important. The method is also applicable when identifying similar Web services.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9
u_1	360	460	390		270		430	460	
u_2	450		400	340	280			450	
u_3		490		410	260	300			380
u_4		?		330		470	420	450	300
u_5	440		280				430	470	
u_6	310		320	430	400				
u_7		390		420	390	450		380	280
u_8		400			370	320	480	390	
u_9		380		?		490	280	480	390

Figure 2: A 9×9 user-item matrix of response time (milliseconds).

3. Prediction Algorithm

In this section, we first present the procedure of UIQPCA and detail its four stages. Then, we analyze the computational complexity of UIQPCA.

3.1. Prediction Procedure. As is shown in Figure 2, given a user u, a Web service i, a set of users $U = \{u_1, u_2, \ldots, u_m\}$, and a set of Web services $I = \{i_1, i_2, \ldots, i_n\}$, the process of UIQPCA, an approach for the prediction of how user u will feel about the quality of Web service i, represented as $q_{u,i}$, consists of four stages, i.e., Stages 1 and 2 in Phase 1 and Stages 3 and 4 in Phase 2:

(1) With reference to U's opinion on the quality of each Web service in I, UIQPCA employs an improved clustering algorithm that follows the minimum covering principal [29] to put similar users and Web services into clusters.

(2) Based on the clustering results, UIQPCA selects a group of users analogous to user u from $U(i)$, denoted as $S(u)$, and a group of Web services

analogous to Web service I selected from $I(u)$, denoted by $S(i)$.

(3) UIQPCA calculates $q^U_{u,i}$, the quality of Web service i, from user u on the basis of $S(u)$. At the same time, UIQPCA also calculates $q^I_{u,i}$, based on $I(u)$.

(4) UIQPCA predicts $q_{u,i}$ by combining $q^U_{u,i}$ and $q^I_{u,i}$.

3.2. Stage 1: Clustering. To identify $S(u)$, UIQPCA first partitions the users in U into different groups according to their opinions on the quality of every individual Web service in I. For every Web service i that belongs to I, UIQPCA maps out the p-dimensional quality of all the users in U who have formerly used i into p-dimensional space, in which every dot means the appraisal of the quality of i by a user. Next, UIQPCA employs clustering algorithm to sort the data dots, which is represented as $D = \{d_1, d_2, \ldots, d_g\}$, $g \le m$, into many groups. The users belonging to the same group are similar to each other and those in varied groups are different from each other. In order to describe this iterative algorithm, we refer to the newly formed group after every iteration as *current cluster* and represent it as C_{cr}. The center of it, namely, c_{cr}, is called the *center of the current cluster*. The radius r_{cr} represents the *radius of the current cluster*. The data dots which do not belong to any group are represented as D_{uc} ($D_{uc} \subset D$).

(1) Determine the centroid of D, represented as $c_D = (\overline{x_1}, \ldots, \overline{x_p})$, where $\overline{x_p} = (\sum_{i=1}^{g} x_{i,p}/g)$ and $x_{i,p}$ is the coordinate of d_i on the p^{th} axis in the space.

(2) Determine the data point closest to c_D in D and let it be the center of the first cluster C_1, denoted by (1) as c_{cr} ($cr = 1$):

$$\min \sqrt{\sum_{i=1}^{p} \left(x_{i,j} - x_{D,j}\right)}, \quad \forall d_i \in D_{uc}, \tag{1}$$

where $x_{D,j}$ is the coordinate of c_D on the j^{th} axis.

(3) Calculate r_{cr} by averaging the distances between c_c and the data points in Duc with the following equation:

$$r_{cr} = \frac{\sum_{d \in D_{uc}} \sqrt{\sum_{j=1}^{p} \left(x_{d,j} - x_{c,j}\right)^2}}{\left|D_{uc}\right|}. \tag{2}$$

(4) Determine the data point furthest from c_{cr} in D_{uc} and let it be the new c_{cr} with

$$\max \sqrt{\sum_{j=1}^{p} \left(x_{i,j} - x_{c,j}\right)^2}, \quad \forall d_i \in D_{uc}. \tag{3}$$

(5) Repeat Steps 3 and 4 until no data point can be identified at Step 4, which means that all data points in D belong to a single cluster, respectively.

To present the pseudocode for the aforementioned clustering algorithm, Algorithm 1 is used. This algorithm is executed on $U(i)$ for every Web service i in I, i.e., a total of n executions. The clustering results are recorded in an $m \times m$ matrix called the *user similarity matrix*, which is represented by M_U. An element represented by $x_{u,v}$ in M_U, $1 \le u, v \le m$, is the total number of times that users u and v were divided into the same cluster in the n executions of the clustering algorithm. After obtaining the result of using the Algorithm 1 to process the user item matrix in Figure 2, the user similarity matrix is shown in Figure 3. Apparently, the user similarity matrix is a symmetric matrix, in which $M_U = M_U^T$, $x_{u,v} = x_{v,u}$, $1 \le u, v \le m$, and $x_{u,v} = x_{v,u} = 0$ if $u = v$.

An illustrative example is presented in Figure 4 to show the clustering process of Algorithm 1. To cluster the data points, Algorithm 1 performed four iterations to identify four clusters, $C_1, \ldots, C_4$, with $c_1, \ldots, c_4$ as the centers and $r_1, \ldots, r_4$ as the radiuses, respectively. Our clustering algorithm can leave out the process of prespecifying the number of clusters and to some degree resist the impact of the preselected starting point(s) like k-means.

Then, we do analysis on the computational complexity of Algorithm 1 based on the 5-step procedure discussed above. At Step 1, the computational complexity is $O(m)$ since there is a maximum of m data points in D. Similarly, the computational complexity of Step 2, Step 3, and Step 4 is also $O(m)$. We need to repeat Step 3 and Step 4 so that all data points in D are processed. At Step 3, the radius of a cluster is the average distances between the cluster center and all data points that are not covered by any clusters. On average, half of the data points uncovered are covered by each newly created cluster. The computational complexity is $O(\log m)$. In this way, the computational complexity of Algorithm 1 is $O(m) \times O(\log m) = O(m \log m)$. For n Web services, Algorithm 1 needs to be executed for n times. So, for all n Web services, the computational complexity of clustering m users is $O(mn \log m)$.

Based on a clustering algorithm analogous to Algorithm 1, according to quality values given by public users, the Web services belonging to I can also be put into different groups by UIQPCA. The clustering algorithm is repeated for m times in total, each for one of the m users. The clustering results are recorded in an $n \times n$ *service similarity matrix* denoted by M_I. An element in this matrix, namely, $y_{i,j}$, ($1 \le i, j \le n$), means the total number of times that Web services i and j were put into the same cluster during the m executions of the clustering algorithm. Like M_U, M_I is also a symmetric matrix. Similar to Algorithm 1, the computational complexity of the algorithm for clustering Web services is $O(mn \log m)$. As a whole, the computational complexity of the whole process of clustering at Stage 1 in Figure 1 is $O(mn \log m) + O(mn \log m) = O(mn \log m)$.

3.3. Stage 2: Selection. After partitioning similar users and Web services into different groups, we choose users similar to user u and Web services similar to Web service i. In order to select, those users and Web services that have the greatest similarity to user u and Web service i, respectively, should be prioritized. The two aspects reflecting the similarity between different users are as follows: (1) common quality experiences, namely, the similarity of their opinions on those Web services that they have both used formerly; (2) common Web

Input: $U(i)$ and i
Output: A set of clusters C_1, C_2, ...
Begin
(1) $cr \longleftarrow 1$;
(2) identify c_D
(3) **do**
(4) identify c_{cr} //(1) when $cr = 1$ and (3) otherwise
(5) Cc.center $\longleftarrow c_{cr}$
(6) calculate r_{cr}//(2)
(7) C_{cr}.radius $\longleftarrow r_{cr}$
(8) $cr \longleftarrow cr + 1$
(9) **while** $(D_u \neq \varnothing)$
End

Algorithm 1: Clustering users $U(i)$ that have invoked web service i.

$$M_U = \begin{array}{c} \begin{matrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8 & u_9 \end{matrix} \\ \begin{bmatrix} 0 & 3 & 0 & 1 & 2 & 1 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 & 2 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 & 0 & 2 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 1 & 0 & 0 & 2 & 1 \\ 2 & 2 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 2 & 0 & 1 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 3 & 0 \\ 1 & 0 & 0 & 2 & 1 & 1 & 3 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{matrix}$$

Figure 3: An example user similarity matrix.

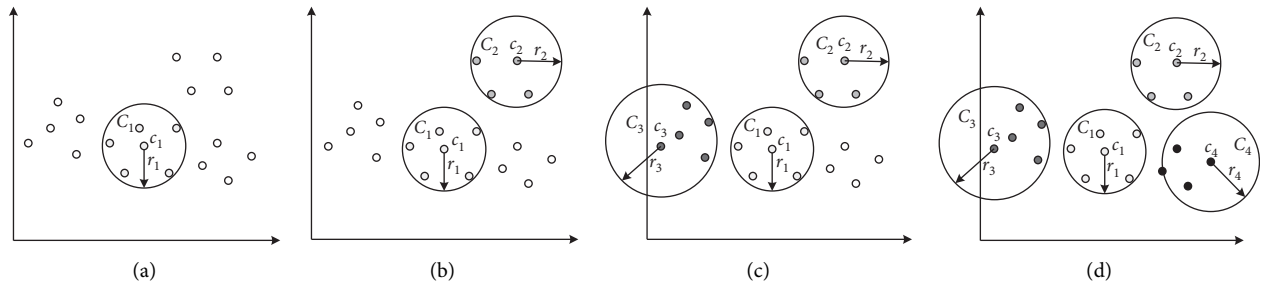

Figure 4: A clustering example. (a) Iteration #1. (b) Iteration #2. (c) Iteration #3. (d) Iteration #4.

services, namely, the number of Web services that they have both used formerly. The algorithm introduced in Section 3.2 has addressed the first issue. Given a Web service i which both user u and user v have used, they will be put into the same group when they have similar opinions on i by executing Algorithm 1 for Web service i. If a Web service has not been used by both the two users, then the Web service will not be included in the process of Algorithm 1. Next, we have to come to the second issue.

If a Web service is used in large amount by both the two users, then its impression on the two users plays an important role in computing their similarity, which acts as a basis of collaborative filtering [22]. The filtering also follows the principle of statistical significance that high statistical significance excludes the impact of chance on the result. For instance, if two users have invocated a good many Web services and have experienced very similar or even the same response time, they will most likely experience similar or even the same response time for other Web services. The fact that they are located at the same regional network and use the same network infrastructure when approaching the same Web services can be a possible but not the sole explanation. Conversely, if the number of Web services used by the both two users is quite limited, then even if they have had similar opinions on the quality of these Web services, it is uncertain that they will react similarly to other Web services. For

instance, if the number of Web services coevoked by two users is as small as one, then their similar response time or quality opinion on this Web service can be caused by accident.

Therefore, while choosing users who are similar to user u, UIQPCA first considers the number of Web services that those users and user u have both used. The similarity of users is determined according to the results from Stage 1. Given a set of users, we execute the clustering algorithm for many times for every coused Web service on the basis of these users' feelings about the quality of the services. That is to say, users having similar experiences at each service can be found through UIQPCA. Suppose that there are two users, u_1 and u_2; if they are always partitioned into a same cluster, it means that they are quite similar in their experiences on all the services, which in turn shows a high similarity between them. Then, we figure out the frequency that the two users are put into a same cluster in Stage 1 in order to analyze their similarity.

According to the aforementioned M_U, namely, the $m \times m$ users' similarity matrix in Section 3.2, containing the results of users clustering, an element $x_{u,v}$, $1 \le u,\ v \le m$, is the total number of times that users u and v have been partitioned into the same cluster. A high value of $x_{u,v}$ shows that both users u and users v have used large number of Web services and have experienced similar quality experiences on those Web services. Take user u_1 in Figure 3 as example. User u_2 should have the highest priority during the selection of users similar to u_1. Then, user u_5 should be in the second place. The priorities of u_4, u_6, and u_8 are lower compared with u_2 and u_5, but when be compared with u_3, u_7, and u_9, they are higher. Different weights are given to different users with reference to their similarity to u_1 in many ways when choosing similar users. For instance, a set of weights, $w_1, w_2, \ldots, w_9$, can be allocated to users $u_2, u_3, \ldots, u_9$, respectively, to indicate their priorities, where $w_2 > w_5 > w_4$, $w_8 > w_3$, w_7, w_9 and $\sum_{i=2}^{9} w_i = 1$. The top $(1 \le k_U \le m-1)$ users that have been divided into the same cluster for the most times can be selected by UIQPCA.

Given a Web service i, UIQPCA uses a similar approach to the one that is used in selecting similar users to select top k_I $(1 \le k_I \le n-1)$ Web services similar to Web service i. To select similar users and similar Web services, use k (k_U or k_I) as a domain-specific parameter. Different applications and datasets are often characterized with their own features and consequently have varied optimal k values. For one thing, if the k value is too small, then not all the useful information on the users can be employed while predicting quality. For another, if the k value is too large, those dissimilar users will be included and thus influence the accuracy of quality prediction. Considering the importance of its accuracy, based on the experiences and/or experiments, the value of k should be set in a specific domain. In Section 4.5.3, the effects of k on the accuracy of predicting will be studied with two datasets which are used in real world.

Now, we discuss the computational complexity of the selection process. In order to select the top k users or Web services, we need to sort the elements of each entry in M_U and M_I. The computational complexity of sorting process depends on the used clustering algorithm. In this paper, the computational complexity of comparison sorting algorithms is used in the worst-case scenario, namely, $O(m \log m)$, for each entry in M_U and M_I. Overall, the computational complexity of selecting similar users and similar Web services at Stage 2 is $O(m^2 \log m + n^2 \log n)$.

3.4. Stage 3: Prediction. After getting the experiences on quality from the selected k_U users similar to user u, namely, $S(u)$, the UIQPCA method executes (4) an approach based on users to make predictions on how user u will feel about the quality of Web service i with a weighted average of the quality experienced by the k_U users:

$$q_{u,i}^{U} = \frac{\sum_{v \in S(u)} x_{u,v} \times q_{v,i}}{\sum_{v \in S(u)} x_{u,v}}, \tag{4}$$

where $x_{u,v}$ is the number of times users u and v have been partitioned into the same cluster regarding the user similarity matrix M_U and $q_{v,i}$ is the quality of Web service i according to user v.

Similar to the user-based approach, QPCA is based on items in predicting how user u will appraise the quality of Web service i using formula (5) with a weighted average of the quality of the selected k_I services which are analogous to Web service i, i.e., $S(i)$:

$$q_{u,i}^{I} = \frac{\sum_{j \in S(i)} x_{i,j} \times q_{u,j}}{\sum_{j \in S(i)} x_{i,j}}, \tag{5}$$

where $x_{i,j}$ is the number of times Web services i and j have been divided into the same cluster obtained from the service similarity matrix M_I and $q_{u,j}$ is the quality of Web service j experienced by user u.

The computational complexity of (4) is $O(k) = O(m)$ since there are a maximum of $m-1$ users in $S(u)$. Likewise, the computational complexity of (5) is $O(n)$. As a whole, the computational complexity of the quality prediction at Stage 3 is $O(m+n)$.

3.5. Stage 4: Combination. Equations (4) and (5) are employed for the prediction of the quality of a Web service based on similar users and similar Web services, respectively. Then, we have to deal with the cases where only a few similar users or similar Web services are available because of the limited scale of the user-item matrix. In these cases, quality prediction by only similar users with (4) or only similar Web services with (5) is likely to have less accuracy for the insufficiency of data in the user-item matrix. If so, we need to use both the results obtained from both the user-based and item-based approaches. To address this issue, we combine (4) and (5) to get prediction results with more accuracy:

$$q_{u,i} = \lambda \times \frac{\sum_{v \in S(u)} x_{u,v} \times q_{v,i}}{\sum_{v \in S(u)} x_{u,v}} + (1-\lambda) \times \frac{\sum_{j \in S(i)} x_{i,j} \times q_{u,j}}{\sum_{j \in S(i)} x_{i,j}}, \tag{6}$$

where parameter λ $(0 \le \lambda \le 1)$ is a parameter determined by the user of UIQPCA that indicates how much $q_{u,i}$ relies on

similar users and similar Web services, i.e., (4) and (5). Parameter λ is a domain-specific parameter. Its optimal value depends on the features of the datasets. In reality, the number of users and Web services that are really similar to the given user and the target Web service may vary significantly from one domain to another. There is no single λ value that can optimize the prediction accuracy for all datasets in all domains. Thus, parameter λ needs to be determined empirically and domain specifically. Different applications and datasets are characterized with their own features. Accordingly, varied λ values are required to achieve optimal prediction results. A rule of thumb is to specify parameter λ according to the ratio of $S(u)$ over $S(i)$. For example, if $|S(u)| \gg |S(i)|$, the λ value should be close to 1.0, putting most weight on similar users. If $|S(u)| \ll |S(i)|$, the λ value should be close to 0, putting most weight on similar Web services. The impact of λ on the accuracy of UIQPCA is studied experimentally with the real-world dataset in Section 4.5.2. In particular, UIQPCA employs $\lambda = 1$, if $S(u) \neq \varnothing$ and $S(i) = \varnothing$ and (6) is equivalent to (4). UIQPCA employs $\lambda = 0$, if $S(u) = \varnothing$ and $S(i) \neq \varnothing$ and (6) is equivalent to (5). To summarize,

$$q(u,i) = \begin{cases} \lambda \times q_U(u,i) + (1-\lambda) \times q_I(u,i), & S(u) \neq 0 \quad \text{and} \quad S(i) \neq 0, \\ q_U(u,i), & S(u) \neq 0 \quad \text{and} \quad S(i) = 0, \\ q_I(u,i), & S(u) = 0 \quad \text{and} \quad S(i) \neq 0, \\ \text{null}, & S(u) = 0 \quad \text{and} \quad S(i) = 0, \end{cases} \tag{7}$$

where null indicates that prediction is not available without similar users and similar Web services.

The complexity of the combination process at Stage 4 is $O(1)$ since it is only a simple addition of the prediction results obtained from Stage 3 using (4) and (5).

3.6. Analysis on Computational Complexity. In this section, we will discuss the computational overhead of UIQPCA with a comparison with quality prediction approaches based on Pearson correlation coefficient (PCC) [14, 15, 19, 21] and k-means [25]. Considering a $m \times n$ user-item matrix which includes the quality values of n services used by m users, the computational complexity of user-based, item-based, and hybrid PCC prediction methods is $O(m^2n)$, $O(mn^2)$, and $O(m^2n + mn^2)$, respectively [14]. The computational complexity of the k-means-based prediction approach is $O(m^2n)$ [25].

The procedure of UIQPCA consists of four steps. As discussed at the end of Sections 3.2–3.5, the computational complexity of the operations at Stages 1, 2, 3, and 4 is $O(mn)$, $O(m^2 \log m + n^2 \log n)$, $O(m + n)$, and $O(1)$ respectively. In consequence, the computational complexity of UIQPCA as a whole is $O(mn \log m) + O(m^2 \log m + n^2 \log n) + O(m + n) + O(1) = O(mn \log m + m^2 \log m + n^2 \log n)$. The details of the experimental studies on the computational complexity of different prediction approaches are demonstrated Section 4.6.

4. Experimental Evaluation

In this section, we analyze how effective (using prediction accuracy to measure) and efficient (using computational overhead to measure) UIQPCA is. Sections 4.1–4.3 describe the dataset, the comparing methods, and the metrics used in the experiments, respectively. Section 4.4 introduces the setup of the experiments. Sections 4.5 and 4.6 present and discuss the effectiveness and efficiency of UIQPCA. In Section 4.7, the potential negative factors affecting the reliability of our evaluation will be discussed.

4.1. Dataset Description. Existing Web service recommendation approaches are based on either service quality or service ratings. Accordingly, we have conducted a series of experiments with the real-world dataset, WS-Dream (https://github.com/wsdream).

4.1.1. WS-Dream. This dataset contains the two-dimensional quality values of 5,825 real-world Web services, which includes response time as well as throughput. The data were gathered from 339 distributed users' 1,974,6754 invocations on those Web services. This dataset has been used in a lot of important and representative research [21, 25] and has become the real dataset in research on quality prediction for Web services. To compare the performance of our method with existing approaches in a fair, objective with reproducible experimental results, we also used the WS-Dream dataset in our experiments, like many other researchers. In the experiments, two $339 \times 5{,}825$ user-item matrices were built based on the records in WS-Dream. The first one has 339 entries, each containing the response time of one of the 339 users when using the 5,825 Web services. The second matrix has the same structure as the first one, but contains the throughput of the Web services experienced by the users.

4.2. Comparing Methods. There are two main methods for personalized recommendation based on collaborative filtering, user-based and item-based filtering [30]. It is significantly important to identify the most similar users or items. Accordingly, we have implemented three prediction methods based on our algorithm introduced in Section 3:

(i) UQPCA (User-based Quality Prediction with Covering Algorithm): this user-based method employs QPCA to cluster users and identifies similar users and then predicts those unshown quality values of unused Web services according to identified similar users.

(ii) IQPCA (Item-based Quality Prediction with Covering Algorithm): this item-based method employs QPCA to cluster Web services, identifying Web services which are similar, and then predicts the unknown quality values of Web services that have not been used according to determined similar Web services.

(iii) UIQPCA (Hybrid User and Item-based Quality Prediction with Covering Algorithm): this method combines UQPCA and IQPCA to predict the missing quality values of Web services on the basis of both similar users and Web services.

To evaluate UIQPCA, we compared UQPCA, IQPCA, and UIQPCA with the following advanced prediction methods:

(i) UPCC (User-based Collaborative Filtering using PCC) [19]: this method identifies similar users using PCC and predicts the missing quality values of Web services that have not been used with the help of identified similar users.

(ii) IPCC (Item-based Collaborative Filtering using PCC) [20]: this method also identifies similar Web services (items) using PCC and predicts the missing quality values of Web services that have not been used according to identified similar Web services.

(iii) UIPCC (Hybrid User and Item-based using PCC) [15]: combining UPCC and IPCC, this method predicts the missing quality values of Web services that have not been used by taking both similar users and similar Web services into consideration.

(iv) TOSEM [31]: this method improves UIPCC with an enhanced PCC for calculating the similarity between users as well as Web services.

(v) NIMF [32]: this method employs a Neighborhood-Integrated Matrix Factorization method for collaborative and web service quality values prediction for personal uses.

(vi) UCAPK [25]: this User-based Credibility Aware Prediction method employs K-means to sort out users, identifying similar users, and then predicts the missing quality values of Web services that have not been used according to similar users.

UCAPK follows the same pattern as our prediction method UQPCA—to group users in order to identify similar users. With the purpose of enabling that the comparison is sufficient, we employed the following two prediction methods as well:

(i) ICAPK: this Item-based Credibility Aware Prediction method uses K-means method to group Web services, finding similar Web services, and predict the quality values of unknown Web services on the basis of similar Web services.

(ii) UICAPK: combining UCAPK and ICAPK, this method makes predictions on those unknown quality values of Web services that have not been used on the basis of both similar users and similar Web services.

4.3. Metrics. We compare the effectiveness of different methods, which is reflected in their accuracy of prediction. We measure the accuracy with two metrics: mean absolute error (MAE) and root mean squared error (RMSE). MAE is defined as follows:

$$\mathrm{MAE} = \frac{\sum_{u,i} |R(u,i) - P(u,i)|}{N}, \tag{8}$$

where $R(u,i)$ denotes the real quality values of service i experienced by user u, $P(u,i)$ represents the predicted quality value, and N is the amount of all predicted quality values. A MAE value indicates the average difference between a prediction result and the corresponding experienced value. All the individual differences are given different weights equally when calculating MAE.

RMSE is defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{u,i} (R(u,i) - P(u,i))^2}{N}}. \tag{9}$$

When calculating RMSE, the individual differences between the prediction result and the corresponding observed values are each squared and then averaged over the sample. The square root of the average is taken as the final result. Since all errors are squared before being averaged, RMSE is quite effective in showing large errors. Therefore, if we want to avoid large prediction errors, RMSE will be a more useful and practicable method.

Both MAE and RMSE vary from 0 to ∞. They are both negatively oriented scores, which means that the lower the value is, the higher the accuracy of predicting is. They are both used widely when doing research on the prediction of Web service [14, 15, 21, 25].

To compare the efficiency of UIQPCA with other methods, we measure the computational overheads of all methods.

4.4. Experimental Setup. We employ the cross validation technique [33] to set up the experiments, which has also been employed in the start-of-the-art research [14, 15, 21, 25]. In an experiment on a user-item matrix built from WS-Dream, we randomly removed a certain amount of entries from the matrix according to the *matrix density*, which is the percentage of unremoved entries in the matrix. Different prediction methods are employed to predict the unknown values in the matrix. The prediction results are then evaluated against the original (and real) values of the removed Web services to obtain the MAE and RMSE for each prediction method. This way, we can compare the prediction accuracy of our methods, UQPCA, IQPCA, and UIQPCA with the other methods.

In the experiments on the WS-Dream dataset, the matrix density is changed from 0.2 to 0.4 in steps of 0.05, λ (see (6)) from 0.1 to 0.9, and k (see Section 3.3) from 1 to 5 for the comparison of the accuracy of the results obtained by

different prediction methods, as well as the impacts of λ and k on UIQPCA. Through this method, the ability of different methods to cope with datasets under different parameter settings that simulate datasets of various characteristics is evaluated extensively.

All experiments are conducted on a machine with Intel Core i7-4970 3.6 GHZ, 16 G RAM, running Ubuntu 14.04 LTS.

4.5. Effectiveness. This section compares the accuracy of predicting of UIQPCA with existing prediction methods under different parameter settings.

4.5.1. Accuracy of Prediction. Table 1 demonstrates the MAE and RMSE achieved by using different prediction approaches in the experiments on the WS-Dream dataset under different matrix density with $\lambda = 0.6$ and $k = 2$. Thus, the presentation and discussion of those results are not included in this section.

As demonstrated by Table 1, UIQPCA, based on similar users and Web services, obtains better accuracy than all eight existing prediction methods for both response time (RT) and throughput (TP) in different matrix density settings. The advantages of UIQPCA significantly surpass the existing approaches. With regard to response time prediction, the average margins are 31.3%, 31.1%, 25.3%, 25.7%, 30.6%, 24.1%, 25.2%, and 19.7% in MAE and 15.3%, 12.9%, 7.7%, 2.5%, 9.7%, 3.7%, 7.9%, and 7.6% in RMSE versus UPCC, IPCC, UIPCC, UCAPK, ICAPK, UICAPK, TOSEM, and NIMF respectively. And for throughput prediction, the average margins are 30.7%, 52.2%, 38.8%, 23.9%, 46.6%, 18.1%, 34.6%, and 3.8% in MAE and 32.3%, 40.0%, 32.5%, 30.7%, 38.4%, 26.8%, 33.2%, and 5.6% in RMSE. As the matrix density increases from 0.2 to 0.4, UQPCA, IQPCA, and UIQPCA's performances increase similarly as the other methods, as indicated by the corresponding decreases from 0.368 to 0.341 in MAE and 1.221 to 1.161 in RMSE.

4.5.2. Impact of λ. The experimental results presented in Section 4.5.1 have demonstrated that by using the information on both similar users and similar Web services, UIQPCA achieves a better prediction accuracy than UQPCA and IQPCA. In this section, the impact of parameter λ on UIQPCA is discussed. As discussed in Section 3.5, when $\lambda = 0$, UIQPCA is equal to IQPCA as it only considers similar Web services. Likewise, while $\lambda = 1.0$, UIQPCA is equal to UQPCA. Thus, the λ value can be varied from 0.1 to 0.9 to evaluate its impact on UIQPCA. In the experiments on the WS-Dream dataset, the matrix density is 0.2 and $k = 2$. The results of other experiments under different parameter settings are similar and thus are omitted.

The results are shown in Figure 5. It demonstrates that the λ value has great influence on the prediction accuracy of UIQPCA. Overall, the prediction accuracy improves with the increase of λ before going up to its optimum. Then, the prediction accuracy reduces in spite of the continuing increase of the λ value. This indicates that it is essentially important to find an accurate and proper λ value. A proper λ value enables the most accurate prediction since it makes sure that the information on users and web services will combine properly. Figure 5 also shows that the optimal λ value is not static. Instead, it varies from one dataset to another. Take RMSE for example. Figures 5(b) and 5(d) show that the RMSE of the prediction results is the lowest when $\lambda = 0.5$ for the WS-Dream dataset. Our findings in the experiments confirm that the optimal λ value is domain-specific, as discussed in Section 3.5.

4.5.3. Impact of k. As discussed in Section 3.3, parameter k is used to determine the numbers of similar users and/or similar Web services to be included in the prediction. Intuitively, the more similar users and/or Web services are considered when making predictions, the more information that can be employed to predict and the more accurate the result will be in consequence. However, this is not necessarily true. As shown in Figure 6, the increase in the k value does not always ensure more accurate prediction. In a word, with the continuing increase of k value, the accuracy of prediction increases from the very beginning until its maximum value and then begins to go down. This tells us that we do not always want to consider as many users and/or Web services as possible to make accurate predictions with UIQPCA. Similar to our findings about λ presented in Section 4.5.2, k must be proper. In the case of a too small k value, the information on similar users and Web services will be insufficient, which is shown in Figure 6 by the reduction in MAE and RMSE in both series of experiments while the k value begins to increase from 1. Secondly, when the k value is too large, users and/or Web services dissimilar to a certain user may be included, which will negatively influence the accuracy of predication to a large extent. Take Figures 6(a) and 6(c) for example. The MAE of the predictions increases from its lowest point at 0.355 to its highest point at 0.376 as k increases from 2 to 6 in the experiments. Furthermore, the optimal k value also varies from one dataset to another. As shown in Figure 6, in the experiments on the WS-Dream dataset, the optimal k values are 2 for MAE and 3 for RMSE. Our experimental findings about parameter k confirm that our analysis of parameter k in Section 3.3—its optimal value is domain-specific and varies from one dataset to another.

4.5.4. Impact of Matrix Density. In Section 4.5.1, for different approaches, the accuracy of prediction is compared when the matrix density settings are different. In this section, the impact of matrix density on UIQPCA is discussed. The experimental results shown in Figure 7 show the great impact of matrix density on the accuracy of prediction by UIQPCA. In all cases, both MAE and RMSE decrease with the increase of matrix density. In a dataset, the higher the matrix density is, the more information on users and Web services can be used for predicting, which in turn guarantees a more accurate result of prediction.

Table 1: Comparison of MAE and RMSE on the WS-Dream dataset ($\lambda = 0.6$, $k = 2$).

Quality dimension	Method	MAE						RMSE					
		Matrix density											
		0.2	0.25	0.3	0.35	0.4	Avg	0.2	0.25	0.3	0.35	0.4	Avg
RT	UPCC	0.518	0.509	0.497	0.484	0.478	0.497	1.445	1.421	1.405	1.391	1.381	1.409
	IPCC	0.580	0.516	0.480	0.457	0.443	0.495	1.472	1.394	1.355	1.328	1.308	1.371
	UIPCC	0.495	0.469	0.452	0.438	0.429	0.457	1.347	1.306	1.286	1.272	1.253	1.293
	UCAPK	0.503	0.482	0.458	0436	0.419	0.459	1.368	1.263	1.192	1.161	1.134	1.223
	ICAPK	0.534	0.515	0498	0.471	0.446	0.492	1.471	1.378	1.321	1.252	1.186	1.321
	UICAPK	0.482	0.469	0.451	0.432	0.415	0.449	1.31	1.261	1.227	1.204	1.188	1.239
	TOSEM	0.489	0.472	0.456	0.439	0.429	0.456	1.353	1.323	1.293	1.261	1.249	1.296
	NIMF	0.454	0.437	0.421	0.412	0.402	0.425	1.151	1.123	1.104	1.087	1.077	1.108
	UQPCA	0.395	0.386	0.379	0.371	0.366	0.379	1.329	1.311	1.305	1.285	1.272	1.299
	IQPCA	0.404	0.396	0.389	0.381	0.373	0.388	1.236	1.227	1.225	1.205	1.179	1.214
	UIQPCA	**0.368**	**0.358**	**0.354**	**0.345**	**0.341**	**0.353**	**1.221**	**1.205**	**1.201**	**1.181**	**1.161**	**1.193**
TP	UPCC	20.646	19.057	18.531	17.768	17.156	18.632	63.459	60.902	60.486	59.159	58.103	60.422
	IPCC	30.322	28.150	26.825	25.427	24.246	26.994	72.673	69.847	68.307	65.870	64.148	68.169
	UIPCC	23.394	21.618	20.740	19.766	19.918	21.087	64.528	61.729	60.696	58.874	57.524	60.670
	UCAPK	18.123	17.735	16.494	16.325	16.069	16.949	60.530	59.474	58.823	58.243	58.118	59.037
	ICAPK	25.095	24.893	24.417	23.564	22.857	24.165	72.031	69.807	67.293	62.451	60.738	66.464
	UICAPK	17.065	16.785	15.706	14.729	14.401	15.737	56.593	56.168	55.895	55.545	55.203	55.881
	TOSEM	22.651	20.643	19.243	18.534	17.645	19.743	65.341	62.625	60.330	59.958	58.027	61.256
	NIMF	14.494	13.688	13.346	12.923	12.589	13.408	41.414	39.325	38.524	37.744	36.542	38.709
	UQPCA	14.483	13.809	13.467	13.065	12.732	13.511	48.519	46.779	46.178	45.201	44.308	46.197
	IQPCA	18.475	17.759	17.357	16.887	16.646	17.424	48.561	46.948	45.848	44.984	44.191	46.106
	UIQPCA	**13.727**	**13.138**	**12.861**	**12.503**	**12.261**	**12.898**	**42.895**	**41.492**	**40.805**	**40.052**	**39.256**	**40.9**

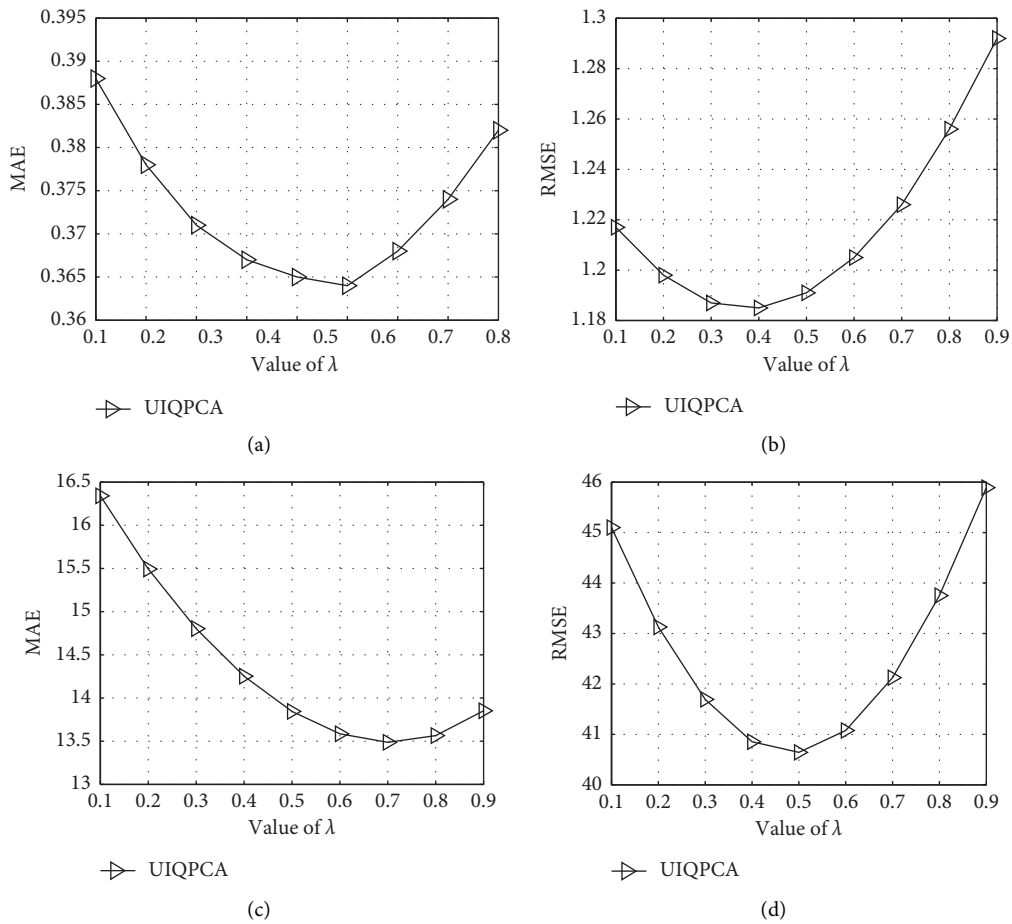

Figure 5: Effects of λ on UIQPCA with the WS-Dream dataset: (a, b) response time; (c, d) throughput. (a) MAE vs. λ. (b) RMSE vs. λ. (c) MAE vs. λ. (d) RMSE vs. λ.

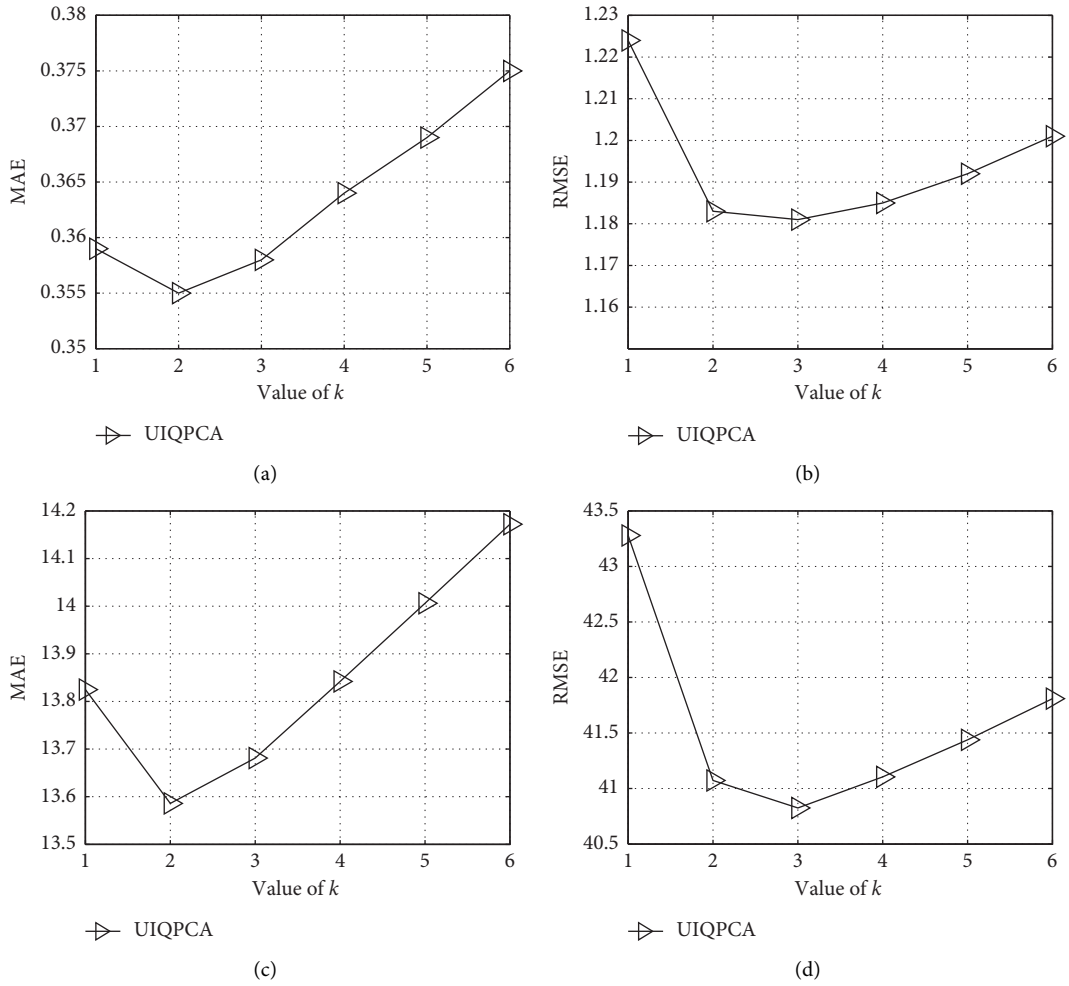

Figure 6: Impact of k on UIQPCA with the WS-Dream dataset: (a, b) response time; (c, d) throughput. (a) MAE vs. k, (b) RMSE vs. k, (c) MAE vs. k, (d) RMSE vs. k.

4.6. Efficiency. Besides the effectiveness measured by prediction accuracy, the efficiency of a prediction method is also critical regarding its feasibility. A prediction method is highly effective; however, the time consumption might not be practical. We have discussed how to determine the complexity of UIQPCA in Section 3.6. To experimentally evaluate the efficiency of UIQPCA, we compare the computation time of UIQPCA against UIPCC, UICAPK, and TOSEM. Those methods, including UIQPCA, consider both similar users and similar Web services when doing predictions and thus are more time consuming than their item-based and user-based versions. Thus, the computation time of their user-based and item-based versions is not presented. The computation time of UIQPCA consists of two major components: *clustering time* and *prediction time*. Here, the clustering component refers to Stage 1 discussed in Section 3.2 and the prediction component refers to Stage 3 discussed in Section 3.4. Accordingly, we compare the clustering time of UIQPCA with that of UICAPK in Figure 8(a), the only comparing method that has a clustering component with k-means. In Figure 8, WSD refers to the WS-Dream dataset, RT refers to response time, and TP refers to throughput. We also compare the prediction time of UIQPCA against UIPCC, UICAPK, and TOSEM in Figure 8(b). In the experiments on the WS-Dream dataset, the parameter settings are matrix density = 0.2, $k = 2$, and $\lambda = 0.6$.

Figure 8(a) shows that UIQPCA spends much less time than UICAPK on clustering the data points in the WS-Dream dataset. Specifically, it only takes 87 milliseconds to cluster the data points in the WS-Dream dataset based on the response time of the Web services, 64% less than UICAPK's 244 milliseconds. Based on throughput, UIQPCA only takes 83 milliseconds, 66% less than UICAPK's 245 milliseconds. The results show that UIQPCA is far more efficient than UICAPK in clustering the datasets. In particular, the clustering of a dataset with UIQPCA is a once-off operation, unless the dataset is updated. After the clustering is completed, the results can be employed to predict any Web services for any users. Thus, UIQPCA is a practical prediction method, especially in large-scale scenarios where a large number of data points need to be clustered.

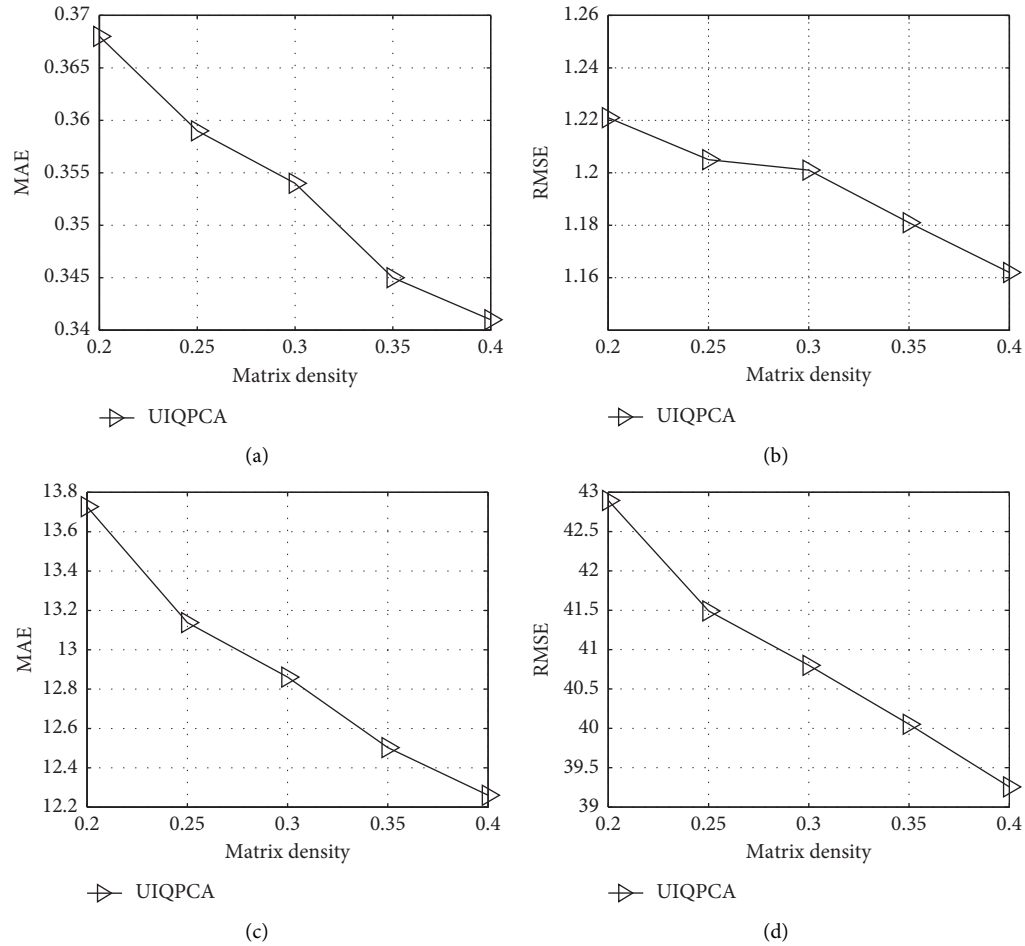

Figure 7: Impact of matrix density on UIQPCA with the WS-Dream dataset: (a, b) response time; (c, d) throughput. (a) MAE vs. matrix density. (b) RMSE vs. matrix density. (c) MAE vs. matrix density. (d) RMSE vs. matrix density.

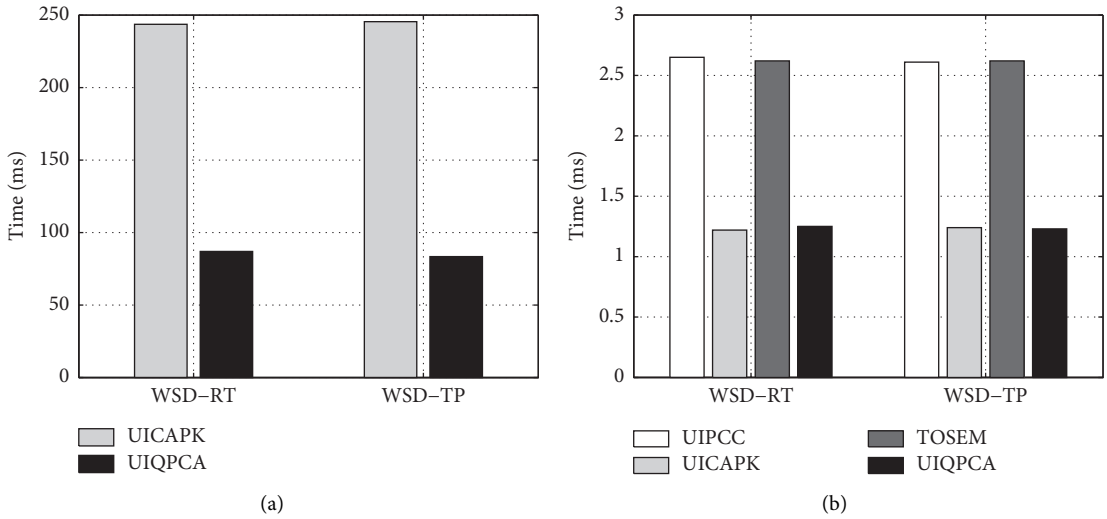

Figure 8: Computation Time. (a) Clustering time. (b) Prediction time.

Figure 8(b) compares the computation time taken by UIQPCA to make predictions against UIPCC, UICAPK, and TOSEM. Overall, all four methods take only 1–3 milliseconds to make a prediction. On average, UIQPCA and UICAPK take a similar amount of time to make a prediction, much less time than UIPCC and TOSEM. On the

WS-Dream dataset, UIQPCA takes approximately half of the time taken by UIPCC and TOSEM to make a prediction. Thus, in scenarios on a much larger scale or when extremely fast predictions are required, UIQPCA is a better choice since it is sufficiently more efficient than UIPCC and TOSEM.

4.7. Threats to Validity. Here, we point out those factors that are possible to negatively influence the results our evaluation of the validity of UIQPCA, including the construct validity, external validity, internal validity, and conclusion validity.

4.7.1. Threats to Construct Validity. The comparison of success rate with the chosen prediction approaches is a major threat to the construct validity of our evaluation. The chosen prediction approaches are conducted on the basis of collaborative filtering technique, which enjoys the greatest popularity and application in current days. There are other methods, e.g., model-based methods [14] and semantics-based methods [13], which are not employed in the evaluation. This threat, however, is not significant since UIQPCA can be compared with those methods in an indirect way by referring to the evaluation shown in relevant literature, e.g., [13, 14], with the approaches used in the evaluation as a reference. The other major threat to the construct validity of our evaluation lies in the insufficiencies of consideration for aspects like timeliness [34] and locations [12] during the prediction process. This threat is also not significant because the consideration of such aspects enhances a prediction method but does not change the fundamental mechanism. The consideration of timeliness and locations can be included in UIQPCA in ways similar to [12, 34]. However, a direct comparison between UIQPCA and other methods in the same category of prediction ways is more direct and typical.

4.7.2. Threats to External Validity. The representativeness of the dataset employed in the evaluation, which might not be as representative and exact when it comes to applications in reality, is the major threat to the external validity in our evaluation. To reduce the effects of this threat to the least, we selected the dataset that is frequently used for experiments on prediction methods for Web services, namely, the WS-Dream dataset [21, 25]. In this way, we could compare UIQPCA with existing methods in a fair and objective manner. In the experiments, we changed the matrix density and parameter λ to simulate datasets with different characteristics and inspected their impacts on UIQPCA. The experimental results demonstrate the effectiveness of UIQPCA on datasets with similar characteristics. This also plays a significant role in reducing the threat to the external validity of our evaluation.

4.7.3. Threats to Internal Validity. The main threat to the internal validity of our evaluation is the comprehensiveness of our experiments. Since the length of this paper is limited, only part of the results of experiments under some parameter settings are presented, which means that we cannot show the results after combining more matrix density and λ values. However, this threat is not believed to be significant. It is possible that the exact values of MAE and RMSE achieved from the experiments under other parameter settings will be different, but the advantages of UIQPCA over the compared methods are similar as presented in the paper.

4.7.4. Threats to Conclusion Validity. The inadequate number of statistical tests—chi-square tests—plays a major role in negatively influencing the validity of conclusion in our evaluation. Chi-square tests were applicable in drawing conclusions when evaluating UIQPCA. But we conducted the experiments for 100 times in each set and averaged the results every time we changed the parameter setting instead, which provided a large amount of test cases and thus tended to lead to a small p value in the chi-square tests and decrease the practical importance of the test results [35]. However, the number of repetition of experiments, when compared with that of the observation samples that concern Lin et al. in [35], is quite small. That is to say, the threat to the conclusion validity caused by the lack of statistical tests might be high in an insignificant way.

5. Related Work

Web service recommendation has been a hot topic of research in recent years. Collaborative filtering-based prediction is currently the mainstream method and has been studied intensively. Over the past years, a lot of researchers have used clustering techniques to improve the collaborative filtering-based prediction methods. Many related works have been done regarding this issue.

5.1. Collaborative Filtering-Based Methods. Collaborative filtering-based methods for Web service recommendation have continued to attract many researchers' attention recently [36–38]. Shao et al. [19] proposed a Web service recommendation method on the basis of collaborative filtering technique. The method first calculates the similarity measured according to PCC between users by taking both their historical feelings about the quality of Web services. Next, it predicts the quality values of the target services on the basis of the information on those users similar to the target user. The approach is implemented in our experiments as UPCC to be compared with UIQPCA. We also implemented a prediction method called IPCC, which is only on the basis of similar Web services. This method was employed by Amazon [39] and has undergone comprehensive investigation and enjoyed great popularity [20]. The principle of this method is to identify similar Web services based on PCC and then make predictions on the unknown quality values of Web services that have not been used on the basis of identified similar Web services.

If predictions are made only on the basis of similar users similar to UPCC under the circumstance that the number of similar users is inadequate, then the result of them can be

unreliable and inaccurate. This also applies to IPCC if the number of similar Web services is inadequate. To solve this problem, Zheng et al. [15] proposes a hybrid prediction method that combines information on both similar users and similar Web services when making predictions. This method identifies not only similar users but also similar Web services according to their similarity measured by PCC. It increases the prediction accuracy to a large extent. Later on, the authors improved their own prediction method presented in [15] with an enhanced similarity computation approach that employs a logistic function to reduce the influence of a small set of Web services coinvoked by dissimilar users [14]. This approach is used as TOSEM in our experiments for comparison with UIQPCA. UIPCC [15] and TOSEM [14] have become the most investigated methods in the field of Web service recommendation. Many researchers have tried to make improvement on them in different ways. To name a few, Zhao et al. [34] put forward a time-aware QoS prediction model which integrates the timeliness of historical QoS data into the process of Web service recommendation. The proposed model cuts the historical QoS data into time slices, each represented by a 2-dimensional matrix to be processed dividedly. Next, it makes predictions on the current QoS values of the target Web services. Yao et al. [13] attempted to unify collaborative filtering and content-based Web service recommendations. Their method considers both the quality values and semantic contents of Web services with the help of probabilistic generative model. Users' preferences for Web services are represented as a set of latent variables, which can be statistically estimated. This way, Web services that are estimated to be preferable to the target user are used to make predictions. Chen et al. [12] proposed a method that employs both the quality values and the location of Web services to make predictions. To estimate a user's location, its IP address is taken into consideration. Similar users are sorted out for their similar opinions on the quality of Web services as well as their locations. This way, a location-aware Web service recommendation system can be built. Wang et al. [37] pointed out a reputation measurement approach, which can avoid the quality comments on web services made by those users who bear biases or malice in their minds. It first identifies those biased and malicious quality ratings by using the cumulative sum control chart and then decreases the influence of subjective users' quality ratings using PCC. The accuracy of prediction can thus be improved.

With the purpose of making improvements on prediction accuracy, the authors of [14, 15] proposed the use of a matrix factorization model to enable the combination of both regional information of similar users and worldwide information of the whole user-service matrix in prediction [32]. This method is implemented as NIMF in experiments for comparison with UIQPCA. Many research studies have been following this piece of work to allow MF-based quality prediction for Web services to accommodate more sophisticated scenarios. For example, Chen et al. proposed an incremental tensor factorization method by considering the temporal information [10]. You et al. proposed a method that models the high-dimensional QoS data as tensors with an important tensor operation for the purpose of predicting the unknown QoS property values [16].

5.2. Clustering-Based Methods. The key to accurate predictions for approaches based on collaborative filtering is to identify similar users and similar Web services. In order to find out similar users and Web services, researchers have tried to sort different users and Web services before making predictions.

Zhu et al. [40] employed the hierarchical clustering technique to put users and Web services into groups with regard to their quality experiences and quality data, respectively. Their prediction approach requires the collection of real-time QoS data, which is not trivial and thus often impractical. Chen et al. [12] also employed the hierarchical clustering technique to put users into different regional groups regarding their locations and opinions on quality. Web services are grouped only on the basis of the similarity in their quality values. Chen et al. [41] employed the agglomerative hierarchical clustering technique to cluster Web services with reference to their historical QoS data. As a result, Web services within the same cluster are under similar physical environments. At the later stage where users similar to the target user are identified, those Web services within the same cluster are treated as one Web service. In this way, the accuracy and efficiency of the prediction can be improved. However, their method ignores the information on similar users. Zhang et al. [42] pointed out an approach that adopts a fuzzy clustering method to cluster users. The approach makes good use of fuzzy clustering as well as PCC. The approach is mainly limited by the significantly important centroids, which are selected at the very beginning and influential to the clustering results. The k-means algorithm has also aroused lots of researchers' interest as a way for clustering users and Web services because it is popular and easy to execute. Yu and Huang proposed CluCF, an approach [26] that clusters users and Web services on the basis of k-means algorithms and pays great attention to reducing the complexity of updating clusters when new users and new Web services are introduced. Similar to [26], Wu et al. [25] also used the k-means algorithm to cluster users, but with a different purpose, namely, to screen out those unreliable users. The prediction method presented in [25] is implemented as UCAPK and extended to ICAPK and UICAPK in our experiments to be compared with our prediction methods. The heavy reliance on the prespecified number of clusters and the centroids selected at the very beginning largely limits the predicting methods which takes k-means as its basis.

The authors of this paper put forward a new method to predict the quality of Web services—UIQPCA—to solve the problems of existing approaches with a unique covering-based clustering technique. It is highly efficient without the need of specifying the number of clusters and initial centroids in advance. The experimental results demonstrate that it is far more effective and efficient than existing prediction methods.

6. Conclusion

The authors of this paper propose UIQPCA as a unique way of quality prediction with the help of covering algorithm. UIQPCA clusters users and Web services based on their opinions on quality and historical quality data, respectively. Given a target Web service for a target user, the similar users and Web services are found out on the basis of clustering results. Then, UIQPCA employs the information on both similar users and Web services to make predictions. The consequences of the experiments on the real-world dataset demonstrate that UIQPCA significantly surpasses existing representative prediction methods, including user-based methods, item-based methods, hybrid methods, and clustering-based methods. It is also demonstrated that UIQPCA surpasses existing methods in integrating and predicating.

Because of the development of cloud computing and big data analytics [43–48], the Web services ecosystem is becoming more complex and large [4, 49–56]. For further studies, we will do research on how to ensure and improve the efficiency of UIQPCA in large-scale scenarios on the Spark platform.

Data Availability

The QoS data used to support the findings of this study can be accessed publicly in the website https://wsdream.github.io/dataset/wsdream_dataset1.html.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61872002) and the Natural Science Foundation of Anhui Province of China (no. 1808085MF197).

References

[1] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[2] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "TOFFEE: task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Transactions on Cloud Computing*, p. 1, 2019.

[3] Y. Chen, J. Huang, C. Lin, and J. Hu, "A partial selection methodology for efficient QoS-aware service composition," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 384–397, 2015.

[4] L. Qi, Q. He, F. Chen et al., "Finding all you need: web APIs recommendation in web of things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[5] Y. Zhang, G. Cui, S. Deng et al., "Efficient query of quality correlation for service composition," *IEEE Transactions on Services Computing*, p. 1, 2018.

[6] Y. Zhang, K. Wang, Q. He et al., "Covering-based web service quality prediction via neighborhood-aware matrix factorization," *IEEE Transactions on Services Computing*, p. 1, 2019.

[7] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing*, p. 1, 2019.

[8] X. Sun, S. Wang, Y. Xia, and W. Zheng, "Predictive-trend-aware composition of web services with time-varying quality-of-service," *IEEE Access*, vol. 8, pp. 1910–1921, 2020.

[9] X. Xu, S. Fu, L. Qi et al., "An IoT-Oriented data placement method with privacy preservation in cloud environment," *Journal of Network and Computer Applications*, vol. 124, pp. 148–157, 2018.

[10] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, "When UAV swarm meets edge-cloud computing: the QoS perspective," *IEEE Network*, vol. 33, no. 2, pp. 36–43, 2019.

[11] L. Qi, X. Zhang, S. Li, S. Wan, Y. Wen, and W. Gong, "Spatial-temporal data-driven service recommendation with privacy-preservation," *Information Sciences*, vol. 515, pp. 91–102, 2020.

[12] X. Chen, Z. Zheng, Q. Yu, and M. R. Lyu, "Web service recommendation via exploiting location and QoS information," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1913–1924, 2014.

[13] L. Yao, Q. Z. Sheng, A. H. H. Ngu, J. Yu, and A. Segev, "Unified collaborative and content-based web service recommendation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 453–466, 2015.

[14] Z. Zheng and M. R. Lyu, "Personalized reliability prediction of web services," *ACM Transactions on Software Engineering and Methodology*, vol. 22, no. 2, pp. 1–25, 2013.

[15] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Transactions on Services Computing*, vol. 4, no. 2, pp. 140–152, 2011.

[16] M. You, X. Xin, W. Shangguang, L. Jinglin, S. Qibo, and Y. Fangchun, "QoS evaluation for web service recommendation," *China Communications*, vol. 12, no. 4, pp. 151–160, 2015.

[17] F. M. Harper, F. Xu, H. Kaur, K. Condiff, S. Chang, and L. Terveen, "Putting users in control of their recommendations," in *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 3–10, ACM, Vienna, Austria, September 2015.

[18] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, p. 19, 2009.

[19] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction for web services via collaborative filtering," in *Proceedings of the 2007 IEEE International Conference on Web Services*, pp. 439–446, IEEE, Salt Lake City, UT, USA, July 2007.

[20] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International World Wide Web Conference*, pp. 285–295, Hong Kong, China, May 2001.

[21] W. Xiong, B. Li, L. He, M. Chen, and J. Chen, "Collaborative web service QoS prediction on unbalanced data distribution," in *Proceedings of the 21st IEEE International Conference on Web Services*, pp. 377–384, IEEE, Anchorage, AK, USA, June 2014.

[22] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: a constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[23] X. Xu, Q. Liu, Y. Luo et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[24] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, p. 1, 2019.

[25] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "QoS prediction of web services based on two-phase K-means clustering," in *Proceedings of the 22nd IEEE International Conference on Web Services*, pp. 161–168, IEEE, New York, NY, USA, June 2015.

[26] C. Yu and L. Huang, "CluCF: a clustering CF algorithm to address data sparsity problem," *Service Oriented Computing and Applications*, vol. 11, no. 1, pp. 33–45, 2017.

[27] X. Zhang, Z. Wang, X. Lv, and R. Qi, "A clustering-based QoS prediction approach for web service selection," in *Proceedings of the International Conference onInformation Science and Cloud Computing Companion*, pp. 201–206, IEEE, Guangzhou, China, December 2013.

[28] Y.-W. Zhang, Y.-Y. Zhou, F.-T. Wang, Z. Sun, and Q. He, "Service recommendation based on quotient space granularity analysis and covering algorithm on Spark," *Knowledge-Based Systems*, vol. 147, pp. 25–35, 2018.

[29] L. Zhang and B. Zhang, "A geometrical representation of McCulloch-pitts neural model and its applications," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 925–929, 1999.

[30] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender Systems Handbook*, pp. 107–144, Springer, Berlin, Germany, 2011.

[31] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: a time-aware personalized QoS prediction framework for web services," in *Proceedings of the 22nd IEEE International Symposium on Software Reliability Engineering*, pp. 210–219, IEEE, Hiroshima, Japan, November 2011.

[32] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2013.

[33] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics Surveys*, vol. 4, pp. 40–79, 2010.

[34] Y. Zhao, Q. Pi, C. Luo, and D. Yan, "CAPred: a prediction model for timely QoS," in *Proccedings of the 2015 IEEE International Conference on Web Services*, pp. 599–606, IEEE, New York, NY, USA, June 2015.

[35] M. Lin, H. C. Lucas Jr., and G. Shmueli, "Research commentary—too big to fail: large samples and the *p*-value problem," *Information Systems Research*, vol. 24, no. 4, pp. 906–917, 2013.

[36] K. Huang, Y. Liu, S. Nepal, Y. Fan, S. Chen, and W. Tan, "A novel equitable trustworthy mechanism for service recommendation in the evolving service ecosystem," in *Proceedings of the 2014 International Conference on Service-Oriented Computing*, pp. 510–517, Paris, France, November 2014.

[37] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, "Reputation measurement and malicious feedback rating prevention in web service recommendation systems," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 755–767, 2015.

[38] J. Yao, W. Tan, S. Nepal et al., "ReputationNet: reputation-based service recommendation for e-science," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 439–452, 2015.

[39] 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, 2003.

[40] J. Zhu, Y. Kang, Z. Zheng, and M. R. Lyu, "A clustering-based QoS prediction approach for web service recommendation," in *Proceedings of the 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pp. 93–98, Shenzhen, China, April 2012.

[41] F. Chen, S. Yuan, and B. Mu, "User-QoS-based web service clustering for QoS prediction," in *Proceedings of the 22nd IEEE International Conference on Web Services*, pp. 583–590, IEEE, New York, NY, USA, June 2015.

[42] M. Zhang, X. Liu, R. Zhang, and H. Sun, "A web service recommendation approach based on QoS prediction using fuzzy clustering," in *Proceedings of the 9th International Conference on Services Computing*, pp. 138–145, IEEE, Honolulu, HI, USA, June 2012.

[43] J. Zhou, J. Sun, P. Cong, Z. Liu et al., "Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT," *IEEE Transactions on Services Computing (TSC)*, p. 1, 2019 in press.

[44] J. Zhou, X. S. Hu, Y. Ma, J. Sun, T. Wei, and S. Hu, "Improving availability of multicore real-time systems suffering both permanent and transient faults," *IEEE Transactions on Computers*, vol. 68, no. 12, pp. 1785–1801, 2019.

[45] J. Zhou, J. Sun, X. Zhou et al., "Resource management for improving soft-error and lifetime reliability of real-time MPSoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 12, pp. 2215–2228, 2019.

[46] S. Zhang, X. Li, Z. Tan, T. Peng, and G. Wang, "A caching and spatial K-anonymity driven privacy enhancement scheme in continuous location-based services," *Future Generation Computer Systems*, vol. 94, pp. 40–50, 2019.

[47] S. Zhang, G. Wang, M. Z. A. Bhuiyan, and Q. Liu, "A dual privacy preserving scheme in continuous location-based services," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4191–4200, 2018.

[48] S. Zhang, K.-K. R. Choo, Q. Liu, and G. Wang, "Enhancing privacy through uniform grid and caching in location-based services," *Future Generation Computer Systems*, vol. 86, pp. 881–892, 2018.

[49] Y. Wang, Z. Cai, Z.-H. Zhan, Y.-J. Gong, and X. Tong, "An optimization and auction-based incentive mechanism to maximize social welfare for mobile crowdsourcing," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 3, pp. 414–429, 2019.

[50] Y. Wang, Z. Cai, G. Yin, Y. Gao, X. Tong, and G. Wu, "An incentive mechanism with privacy protection in mobile crowdsourcing systems," *Computer Networks*, vol. 102, pp. 157–171, 2016.

[51] B. Zhao, Y. Wang, Y. Li, Y. Gao, and X. Tong, "Task allocation model based on worker friend relationship for mobile crowdsourcing," *Sensors*, vol. 19, no. 4, p. 921, 2019.

[52] H. Liu, H. Kou, C. Yan, and L. Qi, "Link prediction in paper citation network to construct paper correlated graph," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, p. 233, 2019.

[53] W. Gong, L. Qi, and Y. Xu, "Privacy-aware multidimensional mobile service quality prediction and recommendation in

distributed fog environment," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 3075849, 8 pages, 2018.

[54] M. D. Ekstrand, D. Kluver, F. M. Harper, and J. A. Konstan, "Letting users choose recommender algorithms: an experimental study," in *Proccedings of the 9th ACM Conference on Recommender Systems*, pp. 11–18, ACM, Vienna, Austria, September 2015.

[55] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, pp. 1–23.

[56] Y. Zhang, C. Yin, Q. Wu et al., "Location-aware deep collaborative filtering for service recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems (TSMC)*, pp. 1–12, 2019.