Covering-based Web Service Quality Prediction via Neighborhood-aware Matrix Factorization

Yiwen Zhang, Kaibin Wang, Qiang He, *Member, IEEE,* Feifei Chen, Shuiguang Deng, *Senior Member, IEEE,* Zibin Zheng, *Senior Member, IEEE,* Yun Yang, *Senior Member*, *IEEE*

Abstract—The number of Web services on the Internet has been growing rapidly. This has made it increasingly difficult for users to find the right services from a large number of functionally equivalent candidate services. Inspecting every Web service for their quality value is impractical because it is very resource consuming. Therefore, the problem of quality prediction for Web services has attracted a lot of attention in the past several years, with a focus on the application of the Matrix Factorization (MF) technique. Recently, researchers have started to employ user similarity to improve MF-based prediction methods for Web services. However, none of the existing methods has properly and systematically addressed two of the major issues: 1) retrieving appropriate neighborhood information, i.e., similar users and services; 2) utilizing full neighborhood information, i.e., both users' and services' neighborhood information. In this paper, we propose CNMF, a **c**overing-based quality prediction method for Web services via **n**eighborhood-aware **m**atrix **f**actorization. The novelty of CNMF is twofold. First, it employs a covering-based clustering method to find similar users and services, which does not require the number of clusters and cluster centroids to be prespecified. Second, it utilizes neighborhood information on both users and services to improve the prediction accuracy. The results of experiments conducted on a real-world dataset containing 1,974,675 Web service invocation records demonstrate that CNMF significantly outperforms eight existing quality prediction methods, including two state-of-the-art methods that also utilize neighborhood information with MF.

Index Terms—quality of service; quality prediction; matrix factorization; neighborhood information; covering algorithm

1 Introduction

The advances in e-Business, eCommerce, especially the pay-as-you-go cloud model have fuelled the rapid growth of Web services [1]. The statistics published by ProgrammableWeb[1], a Web service repository, indicate a rapid growth in the number of published Web services over the past few years. The popularity of Web services and service-oriented architecture (SOA) allows different service-oriented applications to be built to fulfill various organizations' increasingly sophisticated business needs.

To ensure the quality of a service-oriented application, e.g., response time and throughput, it is critical to ensure the quality of the component services that are used to build the service-oriented application. This is a challenging task for three major reasons. First, the quality of a Web service experienced by a user highly depends on both the user's and service's environments. The quality of the same Web service experienced by two different users might be very different [2]. Second, inspecting the quality of every candidate services can be very time-consuming and resource-consuming, especially when the number of candidate services is large [3]. Third, some quality dimensions are difficult to evaluate, e.g., reputation and reliability, because it usually requires tremendous invocations over long observations. Therefore, quality prediction for Web services (referred to as *quality prediction* for short hereafter) has attracted many researchers' attention in recent years and is considered as an effective method for obtaining quality information on Web services.

Collaborative Filtering (CF) has been widely used in quality prediction [4-7]. Existing CF-based prediction methods can be categorized into two major types: memory-based and model-based. Memory-based methods can be further categorized into two subtypes: user-based and item-based CF. To predict the quality of a target Web service i for a target user u, the user-based prediction method needs to first find users similar to user u based on the quality of their co-invoked Web services in the past. Then, it employs the similar users' quality of service (QoS) experiences to make quality prediction for user u. The item-based prediction method follows a similar procedure but employs quality information on services similar to service i to make quality predictions. The

- *Y. Zhang and K. Wang are with the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China. E-mail: zhangyiwen@ahu.edu.cn; wkbahu@qq.com.*
- *Q. He is with the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia and the School of Computer Science and Technology, Anhui University, Hefei, Anhui, China Email: qhe@swin.edu.au.*
- *Feifei Chen is with the School of Information Technology, Deakin University, Melbourne, Australia. E-mail: feifei.chen@deakin.edu.au.*
- *S. Deng is with College of Computer Science and Technology, Zhejiang University, Hangzhou, China. Email: dengsg@zju.edu.cn.*
- *Z. Zheng is with School of Data and Computer Science, Sun Yat-sen University. E-mail: zhzibin@mail.sysu.edu.cn.*
- *Y. Yang is with School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia. Email: yyang@swin.edu.au.*

[1]http://www.programmableweb.com/

Pearson correlation coefficient (PCC) is usually employed to calculate user similarity and service similarity. However, PCC requires an ample amount of quality information. This is often not realistic. In reality, it is very common that a user has invoked only a few services. This lowers the accuracy of the user similarity and service similarity calculated with PCC. Therefore, memory-based prediction methods are vulnerable to data sparsity [4]. A model-based prediction method makes predictions in a different way. It builds a global model based on the available quality information to make quality predictions. In recent years, the Matrix Factorization (MF) technique has been successfully applied to facilitate model-based quality prediction. MF-based prediction is capable of addressing the issue of data sparsity [8] and is currently the most popular quality prediction method [9, 10].

Recently, to improve MF-based quality predictions, researchers are starting to utilize *neighborhood information*, i.e., the quality information provided by users similar to the target user and services similar to the target service [9, 10]. The method named NIMF proposed by Zheng et al. in [9] uses PCC to find similar users in exactly the same way as in their other pieces of work [4-6]. This method, as discussed above, cannot properly address the issue of data sparsity. Tang et al. [10] propose a method named NAMF that finds similar users based on their geographic locations under the assumption that users within the same location have similar QoS experiences. This method limits the utilization of quality information to only those users within the same location and thus does not fully exploit users' neighborhood information based on the quality of their co-invoked services.

Clustering is a promising technique for addressing the issue of data sparsity in memory-based prediction methods. The k-means algorithm has been widely employed to find similar users and similar services [11-13]. There are two major inherent limitations to k-means based quality prediction methods. First, it is difficult to determine a proper k, i.e., the number of clusters, to ensure that the corresponding clustering results can properly represent the distributions of users and services. Second, the clustering results heavily rely on the preselected cluster centroids. Thus, k-means based quality prediction methods cannot ensure the stability and accuracy of the prediction. The other common issue of NIMF [9] and NAMF [10] is that they have not fully utilized the neighborhood information provided by services similar to the target service i. Utilizing only users' neighborhood information, NIMF and NAMF might not be able to make accurate predictions, especially when users' neighborhood information is inadequate.

To address the above issues, this paper proposes CNMF, a **c**overing-based quality prediction method for Web services via **n**eighborhood-aware **m**atrix **f**actorization. CNMF employs a covering-based clustering algorithm that does not require a pre-specified k or cluster centroids to find similar users and similar services. It then makes full use of the neighborhood information by integrating both users' and services' neighborhood information into the MF model to make quality predictions.

The main contributions of this paper are as follows:

1. To obtain appropriate neighborhood information, we employ a covering-based clustering algorithm to partition similar users and similar Web services into clusters effectively and efficiently. This clustering method does not require the number of clusters or the cluster centroids to be pre-specified. This way, CNMF properly addresses the issue of data sparsity and ensures stable quality predictions.
2. To fully utilize neighborhood information, we integrate both users' and services' neighborhood information into the MF model to make quality predictions. This way, CNMF is able to accommodate scenarios where users' or services' neighborhood information can not suffice to ensure the prediction accuracy.
3. To evaluate the performance of CNMF, we perform comprehensive experiments on a real-world dataset that contains a total of 1,974,675 web service invocations all around the world. Experimental results show that CNMF significantly outperforms existing quality prediction methods for Web services, including two state-of-the-art methods that also utilize neighborhood information with MF.

The rest of this paper is organized as follows. Section 2 defines the research problem and motivates this research. Section 3 describes the traditional MF model for quality prediction. Section 4 presents the procedure and details of CNMF. Section 5 presents the experimental settings and results. Section 6 reviews the related work and Section 7 concludes the paper.

2 Problem Definition and Motivation

This section formally defines the research problem and motivates this research.

2.1 Problem Definition

First, we formally define two concepts, namely *neighbor users* and *neighbor Web services*:

Definition 1. Neighbor Users. Given a user u, its neighbor users $N(u)$ are defined as the users that have similar QoS experiences as user u on a set of commonly invoked Web services.

Definition 2. Neighbor Web Services. Given a Web service i, its neighbor Web services $N(i)$ are defined as the Web services that deliver similar QoS experiences as service i to a group of users.

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 procedure for predicting the quality of Web service i consists of two major operations: 1) to identify $N(u)$ from U, and $N(i)$ from I; 2) to predict the quality of Web service i for user u based on $N(u)$ and $N(i)$.

2.2 Motivating Example

	i_1	i_2	i_3	i_4	i_5
u_1	0.2	0.7			1.3
u_2	0.4	0.6			1.1
u_3		0.7	0.7	0.7	
u_4		0.2	0.3		
u_5	0.3		0.3		1.2

Fig. 1. A user-service matrix contains response time (in seconds)

Fig. 1 presents an example with 5 users and 5 Web services. The matrix contains the response time of the Web services experienced by the users. Such a matrix is referred to as a *user-service matrix*, denoted by $M = [q_{ui}]_{m\times n}$, where m is the number of users and n is the number of Web services. Each element q_{ui} in the matrix, $1 \leqslant u \leqslant m$, $1 \leqslant i \leqslant n$, is the quality value of Web service i experienced by user u in its invocation(s) of i in the past. An empty entry represents the unknown quality values. By calculating the PCCs between users based on their QoS experiences in co-invoked Web services, we can find that user u_4 is the user most similar to user u_5 because their PCC is 1.0 based on service i_3, i.e., the only service that they have co-invoked. If we would ignore PCC and take a look at the matrix in general, user u_1 and user u_2 is likely to be more similar to user u_5 than user u_4 because they have both invoked two services with u_5 and share very similar QoS experiences. Let us take a look at another example. Based on PCCs, we can find that user u_3 is most similar to u_1 with a PCC of 1.0. However, it is too soon to conclude that they are so similar because they have only co-invoked one service, i.e., service i_2. Given a sparse user-service matrix, using PCC to find similar users or services might include users and/or services that are in fact not similar in the quality prediction. This will impact the prediction accuracy.

To address this issue, we can cluster users and services to find similar users and services. Take u_5 as the target user for example. Ideally, a properly designed and configured clustering algorithm should be able to assign u_1, u_2 and u_5 to a same cluster so that u_1 and u_2 are selected to be included as u_5's neighbor users in the quality prediction. It is expected that the prediction accuracy will be higher than the one achieved by quality prediction based on u_4 which is selected as the sole similar user based on their PCCs. Here, the key is to assign similar users and/or services into the same clusters, i.e., the clustering quality. The clustering quality achieved by k-means is often not reliable as it heavily relies on k, i.e., the pre-specified number of clusters, and cluster centroids. For example, an overly large k is likely to assign similar users or services into different clusters while an overly small k would probably assign dissimilar users or services into the same cluster. Either way, it will impact the clustering quality, and consequently the prediction accuracy.

Let us now try to predict the quality of service i_4 for user u_4. As analyzed above, the user that is most similar to u_4 is u_5 based on their PCC. Other than user u_5, no users share similar QoS experiences with user u_4. Under such a circumstance, the inclusion of service i_4's neighborhood information might be helpful in predicting the quality of service i_4 for user u_4. Among i_1, i_2, i_3 and i_5, we can find that i_2 and i_3 are similar to i_4 in terms of their QoS experienced by u_3. Including the quality information on services i_2 and i_3 in the quality prediction might increase the prediction accuracy.

3 Matrix Factorization Technique

MF is one of the most popular and effective techniques for predicting missing values in a matrix. MF-based quality prediction employs a factor model to fit the user-service matrix for prediction. It maps both users and services to a joint latent factor space of dimensionality d, such that quality values are modeled as inner products in that space. The premise behind MF is that there are a few potential factors that affect the user's QoS experiences in Web services. Moreover, each potential factor has a great effect on users' QoS experiences, and all potential feature vectors can be constructed through statistical learning theory. The most important step in the MF model is to obtain two independent feature spaces by building an objective function.

We use M to represent the user-service matrix and try to approximate M:

$$M \approx U_{CMF}^{T} \times I_{CMF} \tag{1}$$

where $U_{CMF} \in \mathbb{R}_{d\times m}$ is the latent user feature matrix, $I_{CMF} \in \mathbb{R}_{d\times n}$ is the latent service feature matrix. Vector $\vec{u}$, $1 \leqslant u \leqslant m$, is the latent user feature vector, and vector $\vec{i}$, $1 \leqslant i \leqslant n$, is the latent service feature vector. Their dimensionality is d. A summary of annotations used in this paper can be found in the supplemental material.

Now, the values in matrices U_{CMF} and I_{CMF} need to be estimated. The widely-used objective function below is employed to approximate the original matrix M with U and I by minimizing the following term:

$$\min_{U,I} \frac{1}{2} \sum_{u=1}^{m} \sum_{i=1}^{n} \left\| q_{ui} - \vec{u}\vec{i} \right\|_F^2 \tag{2}$$

where $\|\cdot\|_F$ denotes the *Frobenius norm* [14] that calculates the difference between the real value in M and the corresponding estimated value calculated with $\vec{u}\vec{i}$. Since each user may invoke only a small number of Web services, M is usually very sparse. Therefore, (2) can be modified as follows:

$$\min_{U,I} \frac{1}{2} \sum_{u=1}^{m} \sum_{i=1}^{n} F_{ui} (q_{ui} - \vec{u}\vec{i})^2 \tag{3}$$

where F_{ui} is the indicator function that returns 1 if user u has invoked service i, or 0 otherwise. To obtain the optimal U and I approximations to the original matrix M, and to avoid overfitting, two regularization terms related to U and I respectively are included to transform (3) into (4):

$$\min_{U,I} L(M,U,I) = \frac{1}{2} \sum_{u=1}^{m} \sum_{i=1}^{n} F_{ui} (q_{ui} - \vec{u}\vec{i})^2 + \frac{\lambda_1}{2} \|U\|_F^2 + \frac{\lambda_2}{2} \|I\|_F^2 \tag{4}$$

where λ_1 and λ_2 are two parameters that control the degree of regularization. Objective function L uses a quadratic regular term to minimize the sum of squared errors. Since it is not convex, it is unrealistic to design an algorithm to find the global minimum [15]. Instead, the stochastic gradient descent technique [16] is employed to

find the approximate optimal solution with (5) and (6):

$$\vec{u} = \vec{u} - \gamma_1 \frac{\P L}{\P \vec{u}} \quad (5)$$

$$\vec{i} = \vec{i} - \gamma_2 \frac{\P L}{\P \vec{i}} \quad (6)$$

where $\gamma_1 > 0$ and $\gamma_2 > 0$ represent the learning rates.

4 CNMF

This section presents and discusses CNMF, including its overall procedure and four phases.

4.1 Overall Procedure

Fig. S1 in the supplemental material shows the overall procedure of CNMF, which consists of four phases:

1. Covering-based Clustering. Based on users' QoS experiences, CNMF employs an improvised clustering algorithm that follows the minimum covering principal [17] to partition similar users and Web services into clusters.
2. Neighbor Selection. Based on the clustering results, CNMF calculates the number of times that every two users or services are clustered into a same cluster. Given a target user u and a target Web service i, it selects the top k users and services that are most similar to u and i respectively as their neighbors, denoted by $N(u)$ and $N(i)$ respectively.
3. Matrix Factorization. Given $N(u)$, CNMF integrates a regularization term that minimizes the difference between user u and $N(u)$ into the MF model. This is referred to as *user-integrated matrix factorization*. The outcomes are two matrices, a user-based latent user feature matrix, denoted by $U_{CNMF\text{-}U}$, and a user-based latent service feature, denoted by $I_{CNMF\text{-}U}$, both optimized based on user u's neighborhood information. In the meantime, CNMF does the same for service i. It performs *service-integrated matrix factorization* and generates two matrices that are optimized based service i's neighborhood information, denoted by $U_{CNMF\text{-}I}$ and $I_{CNMF\text{-}I}$. In total, four matrices will be produced in this phase, two latent user feature matrices and two latent service feature matrices.
4. Prediction. Given the four matrices produced by matrix factorization, CNMF takes the corresponding vectors to predict the quality of service i for user u. This is referred to as the *user-based quality prediction* because it utilizes the information provided by $U_{CNMF\text{-}U}$ and $I_{CNMF\text{-}U}$. CNMF also performs *service-based quality prediction* using the vectors in $U_{CNMF\text{-}I}$ and $I_{CNMF\text{-}I}$ that correspond to service i. Finally, CNMF combines the results to make the final prediction.

The complexity analysis of CNMF is presented in the supplemental material.

4.2 Covering-based Clustering

Given a set of users U, a set of services I and a user-service matrix M, CNMF first partitions U into clusters based on their QoS experiences in each service in I. For each Web service $i \in I$, CNMF maps the p-dimensional quality of all the users in U that have invoked i in the past into a p-dimensional space, where each data point in the space represents a user's QoS experience in service i. CNMF then employs the following clustering algorithm to partition the data points denoted by $D = \{d_1, d_2, \ldots, d_g\}$, $2 \leqslant g \leqslant m$, into a number of clusters. Users in the same cluster are similar and the ones in different clusters are dissimilar. In the description of this iterative algorithm, the newly formed cluster at each iteration is referred to as the *current cluster*, denoted by C_{cr}. Its center, denoted by c_{cr}, is referred to as the *center of the current cluster*. Its radius, denoted by r_{cr}, is referred to as the *radius of the current cluster*. The set of data points that are not yet covered by any cluster is denoted by D_{uc} ($D_{uc} \in D$). The following is the clustering process:

1. Identify the centroid of D, denoted by $c_D = (\overline{x_1}, \ldots, \overline{x_p})$, 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. Identify the data point in D that is closest to c_D and use it as the center of the first cluster C_1, denoted by c_{cr} ($cr = 1$) with (7):
$$\min \sqrt{\sum_{j=1}^{p} (x_{i,j} - x_{D,j})^2} \quad \forall d_i \in D_{uc} \quad (7)$$
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_{cr} and the data points in D_{uc} with (8) and obtain C_{cr}:
$$r_{cr} = \frac{\sum_{d \in D_{uc}} \sqrt{\sum_{j=1}^{p} (x_{d,j} - x_{c,j})^2}}{| D_{uc} |} \quad (8)$$
4. Delete the data points in C_{cr} from D_{uc}. Identify the data point in D_{uc} that is furthest from c_{cr} and use it as the new c_{cr} with (9):
$$\max \sqrt{\sum_{j=1}^{p} (x_{i,j} - x_{c,j})^2} \quad \forall d_i \in D_{uc} \quad (9)$$
5. Repeat Steps 3 and 4 until no data point can be identified at Step 4, which means all data points in D are covered by one and only one cluster.

The pseudocode for the above clustering algorithm is presented in Algorithm 1. This algorithm is executed on $U(i)$ for every Web service $i \in I$, i.e., a total of n execu-

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

Input: $U(i)$ and i

Output: A set of clusters C_1, C_2, …

```
Begin
1:  cr ← 1;
2:  identify c_D
3:  do
4:  |  identify c_cr      //(1) when cr = 1 and (3) otherwise
5:  |  C_c.center ← c_cr
6:  |  calculate r_cr      //(2)
7:  |  C_cr.radius ← r_cr
8:  |  cr ← cr + 1
9:  while (D_u ≠ Ø)
End
```

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

Fig. 2. An example user similarity matrix

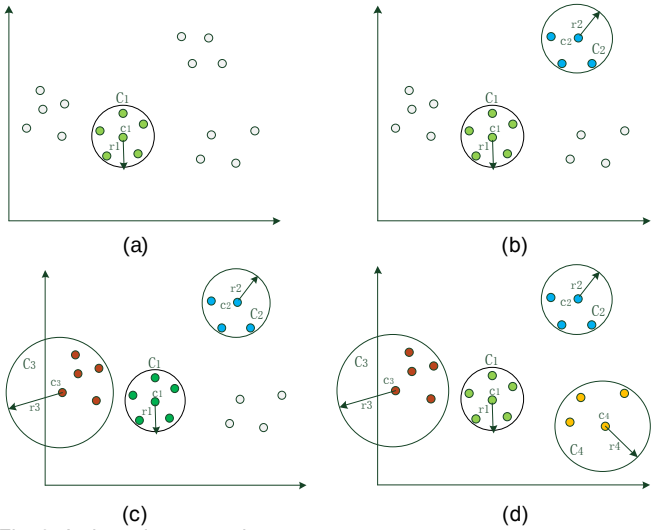


Fig. 3. A clustering example.

tions. The clustering results are recorded in an $m \times m$ matrix named the *user similarity matrix*, denoted by M_U. An element in M_U, denoted by $x_{u,v}$, $1 \leqslant u, v \leqslant m$, is the total number of times that users u and v were partitioned into the same cluster during the n executions of the clustering algorithm. Fig. 2 presents an example user similarity matrix. It is a symmetric matrix. There is $M_u = M_U^T$, $x_{u,v} = x_{v,u}$, $1 \leqslant u, v \leqslant m$ and $x_{u,v} = x_{v,u} = 0$ if $u = v$.

Fig. 3 presents an example to demonstrate the clustering process of Algorithm 1. To cluster those data points, Algorithm 1 goes through four iterations to identify four clusters, C_1, …, C_4, with c_1, …, c_4 as the centroids, and r_1, …, r_4 as the radiuses respectively. Our clustering algorithm does not require the number of clusters or the cluster centroids to be pre-specified like k-means.

4.3 Neighbor Selection

The next phase is to select user u's neighbors, denoted by $N(u)$, and service i's neighbors, denoted by $N(i)$. The selection must prioritize users and services that are most similar to user u and service i. The similarity between two users u and v can be evaluated by addressing two issues: 1) common QoS experiences, i.e., how similar their QoS experiences are in each of the services they have both invoked in the past; 2) common services, i.e., how many services they have both invoked in the past. Algorithm 1 has addressed the first issue. Given a service i that has been invoked by both users u and v, if their QoS experiences in i are similar, they will be partitioned into the same cluster. The services that have not been invoked by both users u and v will not even be considered by Algorithm 1. Now, we need to address the second issue.

When the number of services co-invoked by two users is large, their QoS experiences in those services can significantly contribute to the computation of their similarity. This is a fundamental assumption of collaborative filtering [18]. It also complies with the principle of statistical significance – high statistical significance indicates that the result is not attributed to chance. For example, if two users have invoked a large number of Web services and have experienced highly similar, or even the same, response times, it is highly likely that they will have similar or even the same experiences on the response time of other Web services. A possible (but not necessarily the only) reason is that they might be located within the same local area network and rely on the same network infrastructure when accessing the same Web services. On the contrary, if two users have only co-invoked a very small number of Web services and have had similar QoS experiences, it does not necessarily mean that they will have similar QoS experiences on other Web services. Statistically, the same experiences they had on that only co-invoked Web services could be attributed to coincidence.

Therefore, CNMF prioritizes users that have co-invoked a larger number of Web services with user u when finding $N(u)$. User similarity is evaluated based on the clustering results obtained in the covering-based clustering phase. Given a set of users, the clustering algorithm is executed for multiple times, one for each co-invoked service based on their experiences on the service. This way, CNMF can find users that have similar experiences as per service. For example, if two users are always partitioned into a same cluster, they have very similar experiences on all the services. This indicates a high similarity between them. Thus, we calculate the number of times that two users are partitioned into a same cluster in the covering-based phase to evaluate their similarity. A similar method is applied to evaluate the service similarity.

Let us recall that, in M_U, i.e., the $m \times m$ user similarity matrix introduced in Section 4.2 that contains the results of user clustering, an element $x_{u,v}$, $1 \leqslant u, v \leqslant m$, is the total number of times that users u and v have been partitioned into the same cluster. A high value of $x_{u,v}$ indicates that users u and v have co-invoked a large number of Web services and have had experienced similar QoS on those Web services. Take user u_1 in Fig. 2 for example. User u_2 should have the highest priority during the selection of users similar to u_1. User u_5 should have the second highest priority. Users u_4, u_6 and u_8 should have priorities lower than u_2 and u_5 but higher than users u_3, u_7 and u_9. There are many ways to allocate different weights to different users during the selection of similar users according to their priorities. For example, a set of weights, w_2, …, w_9, can be allocated to users u_2, u_3, …, u_9 respectively to indicate their priorities, where $w_2 > w_5 > w_4$, w_6, $w_8 > w_3$, w_7, w_9 and $\sum_{i=2}^{9} w_i = 1$. CNMF selects the top k_U ($1 \leqslant k_U \leqslant m - 1$) users that have been partitioned into the same cluster with user u for most times.

Given a service i, CNMF employs a method similar to the one for the selection of similar users to select top k_I ($1 \leqslant k_I \leqslant n - 1$) services similar to service i.

For the selection of similar users and similar services, k (k_U for users or k_I for services) is a domain-specific parameter. Different applications and datasets usually have their own characteristics, and hence inherit different optimal k values. On the one hand, a too small k value cannot ensure that all useful information on the users is used to make quality predictions. On the other hand, a too large k value leads to the inclusion of dissimilar users into the quality prediction, and consequently decreases the prediction accuracy. Therefore, the k value should be set domain-specifically based on experiences and/or experiments. In Section 5.5, we study the impact of k on the prediction accuracy with a real-world dataset.

4.4 Matrix Factorization

Given $N(u)$ and $N(i)$, CNMF employs the MF technique to obtain four matrices, i.e., $U_{CNMF\text{-}U}$, $I_{CNMF\text{-}U}$, $U_{CNMF\text{-}I}$ and $I_{CNMF\text{-}I}$. $U_{CNMF\text{-}U}$ and $I_{CNMF\text{-}U}$ are referred to as user-based latent user feature matrix and latent service feature matrix respectively. They are obtained with user-integrated MF. $U_{CNMF\text{-}I}$ and $I_{CNMF\text{-}I}$ are referred to as service-based latent user feature matrix and service-based latent service feature matrix, obtained with service-integrated MF.

User-integrated MF. To perform user-integrated MF, CNMF first calculates a weight for each of user u's neighbors, denoted by $w(u,v)$:

$$w(u,v) = \frac{x_{u,v}}{\sum_{w \in N(u)} x_{u,w}} \tag{10}$$

where $x_{u,v}$ is the number of times that users u and v are assigned into a same cluster in the covering-based clustering phase. A high $w(u,v)$ indicates high similarity between users u and v.

Next, CNMF integrates the following regularization term into the objective function of the prediction model:

$$min \sum_{v \in N(u)} w(u,v) \left\| \vec{u}_u - \vec{v}_u \right\|_F^2 \tag{11}$$

where $\vec{u}_u$ and $\vec{v}_u$, $1 \leqslant u, v \leqslant m$, are the latent user feature vectors from $U_{CNMF\text{-}U}$.

This regularization term has three characteristics: 1) if $w(u,v)$ is large, i.e., the users u and v are similar, user v can contribute more latent information to u; 2) this term is sensitive to neighbors that have diverse QoS experiences; and 3) this term aims to minimize the latent difference between each user and its neighbors to facilitate personalized quality prediction.

The objective function of the user-integrated MF model is now transformed into:

$$\min_{U,I} L'(M,U,I) = \frac{1}{2} \sum_{u=1}^{m} \sum_{i=1}^{n} F_{ui} \left(q_{ui} - \vec{u}_u \vec{i}_u\right)^2 + \frac{\lambda_1}{2} \left\|U\right\|_F^2 + \frac{\lambda_2}{2} \left\|I\right\|_F^2 + \frac{\alpha}{2} \sum_{u=1}^{m} \sum_{v \in N(u)} w(u,v) \left\|\vec{u}_u - \vec{v}_u\right\|_F^2 \tag{12}$$

where $\alpha > 0$ is a tunable parameter controling the importance of (11). Its value is to be specified empirically [10].

Similarly, we employ the gradient descent method to approximate the optimal $U_{CNMF\text{-}U}$ and $I_{CNMF\text{-}U}$ through repetitive local optimization with (13) and (14):

$$\frac{\partial L'}{\partial \vec{u}_u} = \sum_{i=1}^{n} F_{ui} (M_{ui} - \vec{u}_u \vec{i}_u)(-\vec{i}_u) + \lambda_1 \vec{u}_u + \alpha \sum_{v \in N(u)} w(u,v)(\vec{u}_u - \vec{v}_u) \tag{13}$$

$$\frac{\partial L'}{\partial \vec{i}_u} = \sum_{u=1}^{m} F_{ui} (M_{ui} - \vec{u}_u \vec{i}_u)(-\vec{u}_u) + \lambda_2 \vec{i}_u \tag{14}$$

Given (12), (13) and (14), CNMF employs the stochastic gradient descent technique [16] to find the approximate optimal solution by repetitively updating $U_{CNMF\text{-}U}$ and $I_{CNMF\text{-}U}$ with (5) and (6).

Based on the example user-service matrix M presented in Fig. 1, Fig. 4 uses an example to demonstrate the differences between classic MF (CMF) [8], NIMF [9] and our CNMF-U which utilizes only users' neighborhood information. Classic MF does not integrate neighborhood information in its model. Both NIMF and CNMF-U integrates user u's neighborhood information in their models. One of the major differences is that NIMF calculates PCCs to find $N(u)$ and CNMF-U employs the covering-based clustering method to find $N(u)$. All three prediction methods factorize M into a latent user feature matrix and a latent service feature matrix, then obtain a predicted user-service matrix, i.e., M_{CMF}, M_{NIMF} and $M_{CNMF\text{-}U}$.

Let us now focus on the prediction for user u_5. Classic MF utilizes all other users' information to make the predictions because it does not consider u_5's neighborhood information. NIMF utilizes u_4's information to make the predictions because it calculates PCCs and finds that $N(u_5) = \{u_4\}$. CNMF-U utilizes u_1 and u_2's information to make the predictions because it finds $N(u_5) = \{u_1, u_2\}$. Given three predicted matrices, i.e., M_{CMF}, M_{NIMF} and $M_{CNMF\text{-}U}$, we calculate PCCs and find that in M_{NIMF}, u_4 and u_5's QoS experiences are most similar, while in $M_{CNMF\text{-}U}$, u_1, u_2 and u_5's QoS experiences are most similar. Interestingly, in M_{CMF}, no users are particularly similar to u_5. Those observations are consistent with the characteristics of the three prediction methods, however, they do not necessarily indicate which method is superior in achieving high prediction accuracies. Their prediction accuracies will be evaluated and compared with experiments conducted on a real-world dataset in Section 5.

Service-integrated MF. CNMF also performs service-integrated MF. It is very similar to user-integrated MF but integrates service i's neighborhood information into the prediction model instead of user u's. Instead of (10), it uses (15) to calculate a weight for each service in $N(i)$:

$$g(i,j) = \frac{y_{i,j}}{\sum_{k \in N(i)} y_{i,k}} \tag{15}$$

where $y_{i,j}$ represents the number of times that services i and j are assigned to a same cluster in the phase of covering-based clustering.

Instead of (11), the regularization term below is used:

$$\min \sum_{j \in N(i)} g(i,j) \left\| \vec{i}_i - \vec{j}_i \right\|_F^2 \tag{16}$$

Instead of (12), (17) is used as the objective function of the service-integrated MF:

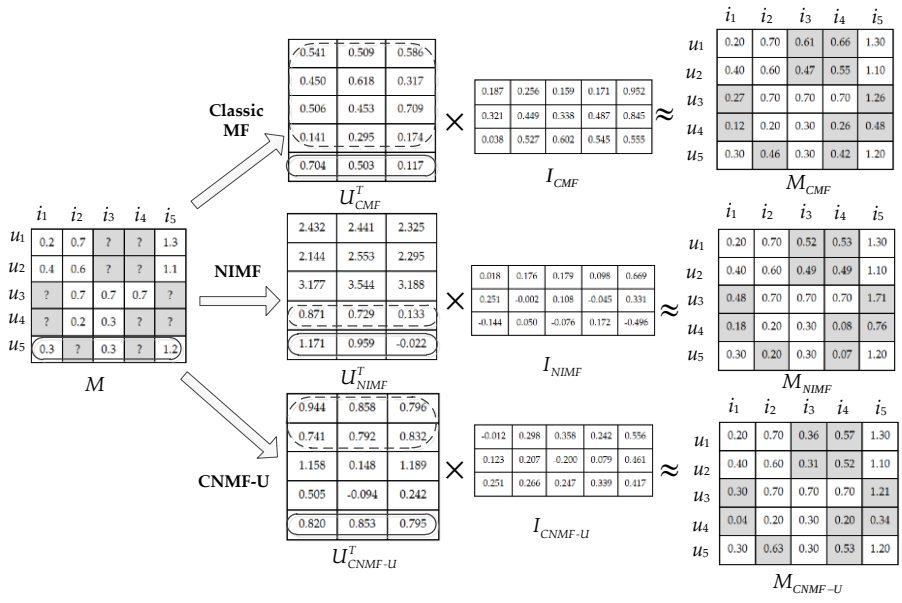


Fig. 4. Prediction with classic MF (CMF), NIMF and CNMF-U.

$$\min_{U,I} L^{"}(M,U,I) = \frac{1}{2}\sum_{u=1}^{m}\sum_{i=1}^{n} F_{ui}(q_{ui} - \vec{u}_i\vec{i}_i)^2 + \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|I\|_F^2 + \frac{\beta}{2}\sum_{i=1}^{n}\sum_{j\in N(i)} g(i,j)\left\|\vec{i}_i - \vec{j}_i\right\|_F^2 \quad (17)$$

Next, service-integrated MF uses (17), (18) and (19) below to approximate optimal $U_{CNMF\text{-}I}$ and $I_{CNMF\text{-}I}$:

$$\frac{\P L^{"}}{\P \vec{u}_i} = \sum_{i=1}^{n} F_{ui}(q_{ui} - \vec{u}_i\vec{i}_i)(-\vec{i}_i) + \lambda_1\vec{u}_i \quad (18)$$

$$\frac{\P L^{"}}{\P \vec{i}_i} = \sum_{u=1}^{m} F_{ui}(q_{ui} - \vec{u}_i\vec{i}_i)(-\vec{u}_i) + \lambda_2\vec{i}_i + \beta\sum_{j\in N(i)} g(i,j)(\vec{i}_i - \vec{j}_i) \quad (19)$$

Finally, the predicted user-service matrix $M_{CNMF\text{-}I}$ is obtained with $U_{CNMF\text{-}I} \times I_{CNMF\text{-}I}$.

4.5 Prediction

Given $U_{CNMF\text{-}U} \times I_{CNMF\text{-}U}$, the user-service matrix predicted based on user u's neighborhood information, denoted by $M_{CNMF\text{-}U}$, is obtained by $U_{CNMF\text{-}U} \times I_{CNMF\text{-}U}$.

Similarly, $M_{CNMF\text{-}I}$, the user-service matrix predicted based on service i's neighborhood information, is obtained by $U_{CNMF\text{-}I} \times I_{CNMF\text{-}I}$.

To make full use of both u's and i's neighborhood information, CNMF combines the predicted values in $M_{CNMF\text{-}U}$ and $M_{CNMF\text{-}I}$ to predict the quality of service i for user u with (20):

$$p(u,i) = \theta p^U_{u,i} + (1-\theta)p^I_{u,i} \quad (20)$$

where parameter θ ($0 \leqslant \theta \leqslant 1$) is employed to determine the degrees in which CNMF depends on user-integrated and service-integrated MF.

5 Experimental Evaluation

This section evaluates the effectiveness (measured by prediction accuracy) of CNMF with comparison to eight existing representative quality prediction methods, including two state-of-the-art MF-based methods that also utilize neighborhood information. The analysis of the threats to the validity of the evaluation is presented in the supplemental material.

5.1 Dataset

The experiments are conducted on a widely-used real-world Web service dataset named WS-Dream[2]. This dataset contains a total of 1,974,675 quality records of 5,825 Web services, generated by the invocations of 339 users distributed across the world. Each record has two quality values, i.e., response time (RT) and throughput (TP). For more details about the dataset, please refer to [19].

5.2 Evaluation Metrics

To evaluate the effectiveness of CNMF, we compare its prediction accuracy measured by Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) with other prediction methods. MAE is defined as:

$$MAE = \frac{\sum_{u,i} |R(u,i) - P(u,i)|}{L} \quad (21)$$

and RMSE is defined as

$$RMSE = \sqrt{\frac{\sum_{u,i}(R(u,i) - P(u,i))^2}{L}} \quad (22)$$

where $R(u, i)$ and $P(u, i)$ are the real and predicted quality value respectively, and L is the total number of predicted quality values.

An MAE value indicates the average difference between a prediction result and the real value. All the individual differences are weighted equally during the calculation of MAE. During the calculation of 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 gives a relatively high weight to large errors. Thus, RMSE is more applicable and useful when large prediction errors are not desirable.

[2] https://github.com/wsdream

TABLE 1
PERFORMANCE COMPARISON

Attributes	Methods	Matrix Density=5%		Matrix Density=10%		Matrix Density=15%		Matrix Density=20%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
RT	UPCC	0.9553	2.1269	0.7823	1.8569	0.6716	1.7264	0.5972	1.7177
	IPCC	1.1026	2.2583	0.8780	1.9893	0.7840	1.8628	0.7223	1.7948
	UIPCC	0.8471	1.9208	0.7290	1.7308	0.6128	1.5906	0.5520	1.5878
	CMF	0.6116	1.4142	0.5169	1.3562	0.4917	1.2163	0.4591	1.1988
	NMF	0.6182	1.5746	0.6040	1.5494	0.5990	1.5345	0.5982	1.5331
	PMF	0.5678	1.4735	0.4996	1.2866	0.4720	1.2163	0.4492	1.1828
	NIMF	0.5514	1.4075	0.4854	1.2745	0.4534	1.1980	0.4357	1.1678
	NAMF	0.5384	1.3853	0.4850	1.2592	0.4529	1.2071	0.4350	1.1443
	KNMF	0.5484	1.3982	0.4849	1.2464	0.4359	1.1653	0.4227	1.1207
	CNMF	**0.5289**	**1.3053**	**0.4713**	**1.2373**	**0.4316**	**1.1360**	**0.4136**	**1.1161**
TP	UPCC	56.4816	95.4345	47.3569	78.1629	41.6976	70.9251	37.2768	67.9981
	IPCC	46.5634	79.6976	42.5893	73.5783	36.5033	68.4784	34.3576	65.4433
	UIPCC	40.0451	74.5033	36.4308	64.9208	33.8068	59.5171	29.2445	56.5301
	CMF	30.8275	69.2836	26.4586	59.3657	21.7944	52.9409	17.8588	47.1635
	NMF	25.7529	65.8517	17.8411	53.9896	15.8939	51.7322	15.2516	48.6330
	PMF	19.9034	54.0508	16.1755	46.4439	15.0956	43.7957	14.6694	42.4855
	NIMF	17.9297	51.6573	16.1755	46.4439	15.0956	43.1596	13.7099	41.1689
	NAMF	18.0837	52.8658	15.9808	44.0788	14.6661	43.0206	13.9386	40.7481
	KNMF	17.6185	52.5914	15.0309	43.6781	14.0120	42.1381	13.9877	40.9735
	CNMF	**17.2036**	**50.6829**	**14.7927**	**43.6639**	**13.9296**	**41.9729**	**13.3993**	**40.0503**

Both MAE and RMSE range from 0 to ∞. A lower MAE or RMSE value indicates a higher prediction accuracy. They have been used widely in research on Web service prediction [20-22].

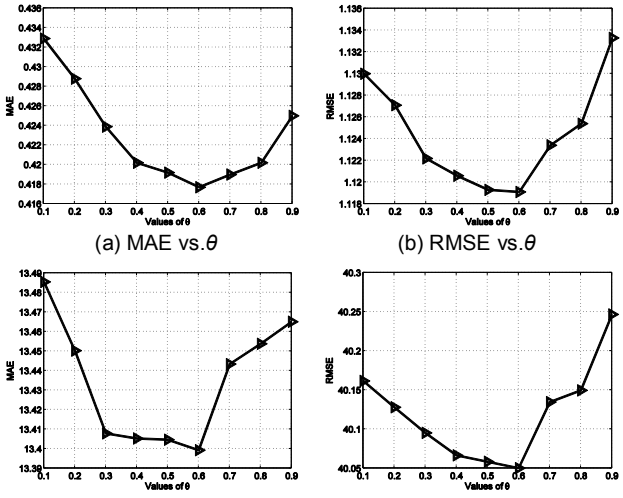
5.3 Comparing Methods

In this section, our method is compared with the following methods in order to demonstrate the accuracy of the prediction. The comparison methods are as follows:

1. User-based CF using PCC (UPCC) [2]. This method identifies $N(u)$ using the PCC and predicts the quality values based on $N(u)$.
2. Item-based CF using PCC (IPCC) [23] . This method identifies $N(i)$ also using PCC and predicts quality values based on $N(i)$.
3. Hybrid user-based and item-based CF using PCC (UIPCC) [24]. This method makes quality prediction based on $N(u)$ and $N(i)$ combined.
4. CMF (classic matrix factorization) [8]. This method is the classic matrix factorization that builds a global model based on the available quality information to make quality predictions.
5. NMF (nonnegative matrix factorization) [25]. This MF-based method makes quality predictions without considering neighborhood information. It requires nonnegative factorized factors.
6. PMF (probabilistic matrix factorization) [26]. This MF-based method differs from other MF-based methods in that it is based on probabilistic matrix factorization, which introduces probability model to further optimize matrix factorization model.
7. NIMF [9]: This method was the first one that integrates users' neighborhood information in making MF-based quality predictions. It calculates PCCs to identify $N(u)$.
8. NAMF [10]: This method also integrates users' neighborhood information in making quality predictions. Unlike NIMF, it identifies $N(u)$ based on their geographic locations.
9. KNMF: The method is the k-means version of CNMF. It employs k-means to identity $N(u)$ and $N(i)$. When running KNMF, the number of clusters obtained by our covering-based algorithm is fed to k-means. This way, their performance can be evaluated without the uncertainty caused by manually-selected values for k.

5.4 Experimental Setup

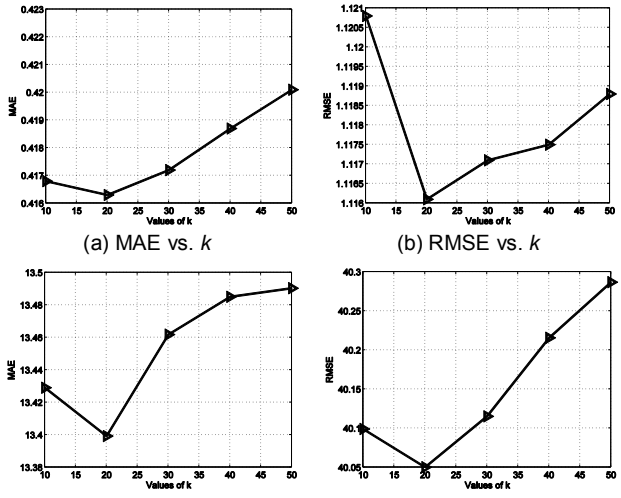
To simulate various prediction scenarios with different data sparsity, we randomly remove certain numbers of quality values from the dataset to generate user-service matrices with 5%, 10%, 15% and 20% densities. For example, a matrix with a 20% density means that we randomly select 20% of the quality values in the original user-service matrix to predict the remaining 80% of the quality values. The removed original quality values are used as the real quality values, i.e., ground truth, to evaluate the prediction accuracies achieved by different methods.



(a) MAE vs.θ (b) RMSE vs.θ

(c) MAE vs. θ (d) RMSE vs. θ

Fig. 5. Impact of θ on CNMF: (a) and (b) response time; (c) and (d) throughput.



(a) MAE vs. k (b) RMSE vs. k

(c) MAE vs. k (d) RMSE vs. k

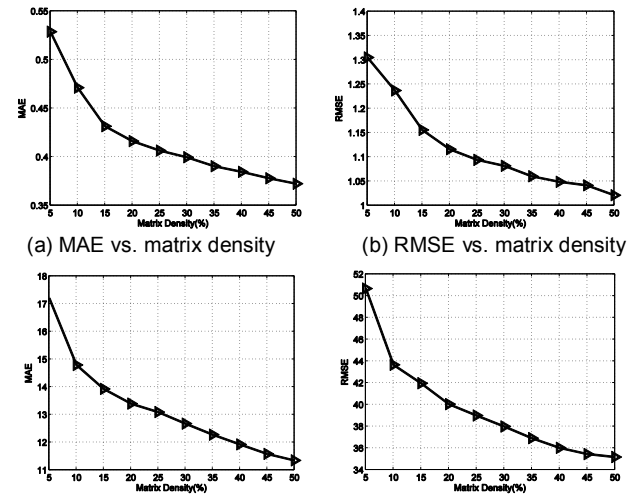
Fig. 7. Impact of k on CNMF: (a) and (b) response time; (c) and (d) throughput.

To conduct an objective and fair comparison, we employ the similar setting for experiment parameters shared with [9, 10], which also integrate neighborhood information in the MF model: $\lambda_1 = \lambda_2 = 0.001$, $\alpha = \beta = 0.001$, $\gamma_1 = \gamma_2 = 0.01$. The maximum number of iterations during matrix factorization is also set at 300. All approaches are implemented in Python 3.6. All experiments are performed on a machine with Intel i7-4790 3.60GHz CPU, 4GB RAM, running Ubuntu 16.04.

5.5 Effectiveness

The section compares the prediction accuracies achieved by CNMF with those by the comparing methods under different parameter settings.

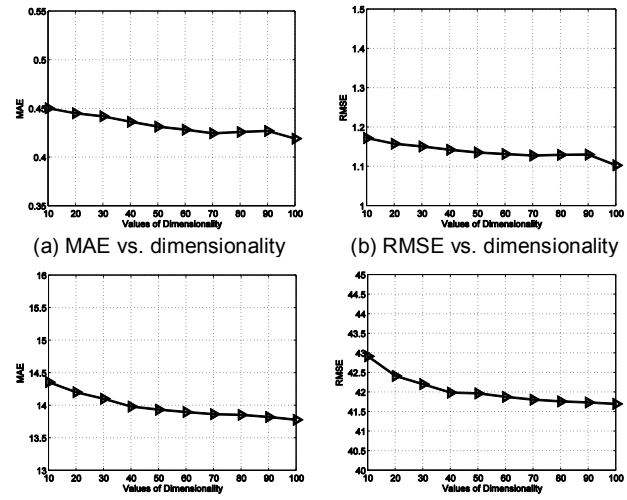
Prediction Accuracy. Table 1 presents MAE and RMSE achieved by different prediction methods under different matrix densities with $\theta = 0.6$ and $k = 20$ as a general comparison. It shows that CNMF achieves the lowest MAE and RMSE across different cases, which demonstrates that CNMF has the best prediction accuracy overall. The advantages of CNMF over existing methods are significant. For response time prediction, the average margins are



(a) MAE vs. matrix density (b) RMSE vs. matrix density

(c) MAE vs. matrix density (d) RMSE vs. matrix density

Fig. 6. Impact of matrix density on CNMF: (a) and (b) response time; (c) and (d) throughput.



(a) MAE vs. dimensionality (b) RMSE vs. dimensionality

(c) MAE vs. dimensionality (d) RMSE vs. dimensionality

Fig. 8. Impact of dimensionality on CNMF: (a) and (b) response time; (c) and (d) throughput.

38.6%, 47.1%, 32.7%, 11.1%, 23.7%, 7.2%, 4.2% , 3.4% and 2.4% in MAE and 35.5%, 65.0%, 29.8%, 7.5%, 22.6%, 7.1%, 5.0%, 4.0% and 2.6% in RMSE in comparison with UPCC, IPCC, UIPCC, CMF, NMF, PMF, NIMF, NAMF and KNMF respectively. For throughput prediction, the average margins are 67.2%, 62.9%, 57.5%, 37.4%, 20.6%, 9.9%, 5.7%, 5.3% and 2.2% in MAE and 43.6%, 38.6%, 31.0%, 22.3%, 19.9%, 5.6%, 3.3%, 2.4% and 1.6% in RMSE. Memory-based CF methods, including UPCC, IPCC and UIPCC, achieve the worst prediction accuracies. In particular, in the cases of very sparse data, MF-based methods outperform memory-based CF methods significantly. As the matrix density increases, the performance of all methods increases. This indicates that more quality information can contribute to higher prediction accuracies. The significant advantage of CNMF over other methods shows that it is more effective in retrieving and utilizing neighborhood information.

Impact of θ. CNMF uses parameter θ to control how much it relies on users' and services' neighborhood information. When $\theta = 0$, CNMF only integrates services' neighborhood information into the prediction. If $\theta = 1$,

CNMF only utilizes users' neighborhood information. It is a domain-specific parameter and needs to be specified empirically or experimentally. To evaluate the impact of parameter θ on the prediction accuracy of CNMF, we change its value from 0.1 to 0.9 with $k = 20$, matrix density = 20% and $d = 50$. Fig. 5 shows the results. We can see that, as θ starts to increase from 0.1, the prediction accuracy increases. When θ reaches 0.6, the highest prediction accuracy is obtained. After that, the prediction accuracy starts to decrease. This shows that for this dataset in this scenario, 0.6 is the optimal value for θ. Accordingly, in the following subsections, we discuss the results achieved by CNMF with $\theta = 0.6$.

Impact of Matrix Density. To further evaluate the impact of matrix density on CNMF, we vary the density of the user-service matrix used in the experiments from 5% to 50% in steps of 5%, with $k = 20$, $d = 50$. The results are shown in Fig. 6. It shows that when the matrix density increases, the MAE and RMSE values both decrease, indicating that the prediction accuracy increases. This shows the importance of ample input data for quality prediction. In particular, Fig. 6 shows that the decrease in both MAE and RMSE is more significant at the beginning of the increase in matrix density. This shows the remarkable advantage of CNMF in handling the issue of data sparsity.

Impact of *k*. The value of k determines the number of neighbor users and neighbor services utilized for prediction. To study the impact of k, we change the value of k from 10 to 50 in steps of 10 with $d = 50$ and matrix density = 20%. Fig. 7 shows the results. It can be seen that as k increases, the prediction accuracy first increases, reaches its optimum at $k = 20$, then decreases. At the beginning, the increase in k will include more similar neighbor users and services in the prediction and increase the prediction accuracy. However, as k continues to increase, some users and services that are in fact not quite similar to user u and service i are included in the prediction. This lowers the prediction accuracy.

Impact of *d*. The dimensionality d determines how many factors are used to factorize the user-service matrix M. To study the impact of d, we change its value from 10 to 100 in steps of 10 for both user-integrated and service-integrated MF, with $k = 20$ and matrix density = 15%. The results are demonstrated in Fig. 8, which shows that the prediction accuracy slowly increases as d increases. This is consistent with the observations discussed in [9].

6 Related Work

With the popularity of service computing technology, quality plays an important role in service selection [27, 28], service discovery [12], service composition [10, 29], and so on. Most of previous studies have assumed that the quality values of candidate services are known and accurate, which is not realistic in most real-world scenarios. Therefore, quality prediction is very important for service computing. Recently, CF-based prediction and MF-based prediction are the mainstream methods for quality prediction for Web services.

6.1 CF-based Methods

User-based CF was first applied to solve this problem by Shao et al. [2]. Then, Zheng et al. [24] proposed WSRec, a hybrid method that combines user-based CF and item-based CF to predict the quality of services. In order to further improve the prediction accuracy, some more sophisticated CF-based quality prediction methods were proposed. They can be roughly divided into two categories. The approaches in the first category attempt to dig deeper into the user-service matrix so that more information can be utilized to make predictions. For example, Jiang et al. [30] discovered personalized features of users and services by analyzing historical QoS records and combined them into the calculation of user and service similarity. Wu et al. [6] employed the data smoothing technique and an improved CF approach to alleviate the issue of data sparsity. IPCC, UPCC and UIPCC are implemented in our experiments as representatives of this category of prediction methods. The second category of approaches employ additional user and service information to assist with prediction. For example, Zhang et al. [31] proposed to use the users' context information for making predictions. Chen et al. [22, 32] used the users' IP addresses to find similar users. They assumed that users with similar IP addresses are located in the same region and have similar QoS experiences. Lo et al. [20] employed users' geographic location information to identify similar users. However, their distance functions are not adequately accurate in capturing the characteristics of the Internet, and thus cannot ensure the prediction accuracy. Tang et al. [21] combine users' location and services' location to find similar users and similar services.

The above methods calculate PCCs to identify similar users and services. However, in the real world, the user-service matrix can be very sparse. As discussed and demonstrated in Section 2, in such scenarios, PCC cannot accurately indicate the user similarity and service similarity, and thus often leads to inaccurate prediction results. Due to its simplicity and popularity, the k-means algorithm has been employed by many researchers to cluster users and services to identify their neighbors. Yu et al. proposed an approach [13] named CluCF that employs the k-means algorithm to cluster users and services. Similar to [13], Wu et al. [11] also employed the k-means algorithm to cluster users, however with a different aim to identify untrustworthy users. The common limitation to prediction methods based on k-means is that the clustering quality heavily relies on the pre-specified number of clusters and the initially selected cluster centers.

6.2 MF-based Methods

To address the issue of data sparsity, matrix factorization, the other widely employed prediction technique in recommendation systems in addition to CF [8], has been employed to make quality predictions for Web services. Zheng et al. [9] proposed a MF-based prediction method that integrates users' neighborhood information into the prediction model. Nonetheless, this method uses PCC to calculate the similarity between users, which is not suitable as discussed in Section 2. Lo et al. [20] also integrated

location-based users' neighborhood information in the MF-based prediction model. They assume that users in a near geographic area tend to share similar QoS experiences in Web services. However, the major reason for the regularization in the MF model is to avoid overfitting in the learning process. Therefore, the method in [20] does not suffice to give a convincing explanation for the contribution of neighbors' QoS experiences. Xu et al. [33] adopted a similar method, which finds the neighbors for users within a distance threshold. The distance between two users is calculated based on their longitudes and latitudes. However, two users that are physically close are not necessarily close in the network. Tang et al. [10] proposed a network-aware MF-based prediction method that integrates the network map into the prediction model. It measures the network distances between users using the network map and then identifies users' neighbors. However, the model treats the neighbors with equal importance and calculates user similarity without considering their historical QoS experiences. In addition, a common and major limitation of the existing MF-based prediction methods is the lack of consideration of services' neighborhood information. CMF [8], NMF [25], PMF [26], NIMF [9] and NAMF [10] are implemented in our experiments as the representative MF-based prediction methods for comparison with our CNMF.

Data sparsity is a major obstacle to retrieving appropriate information for making quality predictions for Web services. To make full use of the neighborhood information, services' neighborhood information also needs to be integrated in the quality prediction. To properly address those two issues, this paper proposes CNMF that employs a covering-based algorithm to identify both users' and services' neighbors to retrieve their neighborhood information. It does not require the number of clusters or the cluster centers to be pre-specified. In addition, to fully utilize the neighborhood information, CNMF integrates both users' and services' neighborhood information into the MF-based prediction model. The experimental results presented in Section 5 demonstrate that CNMF significantly outperforms existing representative quality prediction methods for Web services.

7 Conclusion and Future Work

In this paper, we propose CNMF, a covering-based method for Web services that integrates users' and services' neighborhood information to make quality predictions. CNMF is able to retrieve appropriate neighborhood information by properly overcoming the limitations of user similarity and service similarity calculation based on Pearson correlation coefficient and k-means. Compared with other methods that also integrate users' neighborhood information, it takes a step further to integrate also services' neighborhood information. The results of extensive experiments conducted on a real-world dataset show that CNMF significantly outperforms state-of-the-art quality prediction methods for Web services.

In our future work, we aim to further investigate and improve the covering-based clustering algorithm to improve the accuracy of neighbor identification.

Acknowledgment

This work is partly supported by the National Natural Science Foundation of China (No. 61872002, 61772461), Australian Research Council Discovery Project (DP180100212) and the Natural Science Foundation of Anhui Province of China (No. 1808085MF197). Qiang He is the corresponding author of this paper.

References

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50-55, 2008.

[2] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction forweb services via collaborative filtering," *Proc of Web Services, 2007. ICWS 2007. IEEE International Conference on*, pp. 439-446, 2007.

[3] Q. He, J. Han, Y. Yang, H. Jin, J.-G. Schneider, and S. Versteeg, "Formulating cost-effective monitoring strategies for service-based systems," *IEEE Transactions on Software Engineering*, no. 1, p. 1, 2013.

[4] 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.

[5] H. Sun, Z. Zheng, J. Chen, and M. R. Lyu, "Personalized web service recommendation via normal recovery collaborative filtering," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 573-579, 2013.

[6] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 2, pp. 428-439, 2013.

[7] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 686-699, 2016.

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

[9] 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.

[10] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 126-137, 2016.

[11] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "Qos prediction of web services based on two-phase k-means clustering," *Proc of 2015 IEEE International Conference on Web Services (ICWS)*, pp. 161-168, 2015.

[12] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for web service recommendation," *Knowledge-Based Systems*, vol. 115, pp. 55-65, 2017.

[13] 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.

[14] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM review,* vol. 52, no. 3, pp. 471-501, 2010.

[15] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data (TKDD),* vol. 4, no. 1, p. 1, 2010.

[16] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, ed: Springer, 2010, pp. 177-186.

[17] 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.

[18] 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.

[19] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE transactions on services computing,* vol. 7, no. 1, pp. 32-39, 2014.

[20] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service qos prediction with location-based regularization," *Proc of Web services (ICWS), 2012 IEEE 19th international conference on*, pp. 464-471, 2012.

[21] M. Tang, Y. Jiang, J. Liu, and X. F. Liu, "Location-aware collaborative filtering for QoS-based service recommendation," *Proc of 2012 IEEE 19th International Conference on Web Services*, pp. 202-209, 2012.

[22] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," *Proc of Web Services (ICWS), 2010 IEEE International Conference on*, pp. 9-16, 2010.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," *Proc of Proceedings of the 10th international conference on World Wide Web*, pp. 285-295, 2001.

[24] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," *Proc of Web Services, 2009. ICWS 2009. IEEE International Conference on*, pp. 437-444, 2009.

[25] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature,* vol. 401, no. 6755, p. 788, 1999.

[26] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Proc of Advances in neural information processing systems*, pp. 1257-1264, 2008.

[27] Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin, "QoS-driven service selection for multi-tenant SaaS," *Proc of Cloud computing (cloud), 2012 ieee 5th international conference on*, pp. 566-573, 2012.

[28] Q. He, J. Yan, H. Jin, and Y. Yang, "Quality-Aware Service Selection for Service-Based Systems Based on Iterative Multi-Attribute Combinatorial Auction," *IEEE Trans. Software Eng.,* vol. 40, no. 2, pp. 192-215, 2014.

[29] S. Deng, H. Wu, D. Hu, and J. L. Zhao, "Service selection for composition with QoS correlations," *IEEE Transactions on Services Computing,* vol. 9, no. 2, pp. 291-303, 2016.

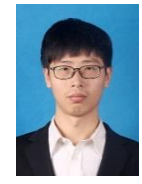
[30] Y. Jiang, J. Liu, M. Tang, and X. F. Liu, "An effective web service recommendation method based on personalized collaborative filtering," *Proc of 2011 IEEE International Conference on Web Services*, pp. 211-218, 2011.

[31] L. Zhang, B. Zhang, Y. Liu, Y. Gao, and Z. L. Zhu, "A Web service QoS prediction approach based on collaborative filtering," *Proc of Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pp. 725-731, 2010.

[32] 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.

[33] Y. Xu, J. Yin, W. Lo, and Z. Wu, "Personalized location-aware QoS prediction for web services using probabilistic matrix factorization," *Proc of International Conference on Web Information Systems Engineering*, pp. 229-242, 2013.

Yiwen Zhang received his PhD degree in management science and engineering in 2013 from Hefei University of Technology. He is an associate professor in the School of Computer Science and Technology at Anhui University. His research interests include service computing, cloud computing, and e-commerce.



Kaibin Wang received his bachelor degree in computer science and technology in 2015 and now is a master student in the School of Computer Science and Technology at Anhui University. His current research interests include service computing and cloud computing.

Qiang He received his first PhD degree from Swinburne University of Technology, Australia, in 2009 and his second PhD degree in computer science and engineering from Huazhong University of Science and Technology, China, in 2010. He is a senior lecturer at Swinburne. His research interests include service computing, software engineering, cloud computing and edge computing. More details about his research can be found at https://sites.google.com/site/heqiang/.

Feifei Chen received her PhD dsegree from Swinburne University of Technology, Australia in 2015. She is a lecturer at Deakin University. Her research interests include software engineering, cloud computing and green computing.

Shuiguang Deng is a full professor at the College of Computer Science and Technology in Zhejiang University. He received the BS and PhD both in computer science from Zhejiang University in 2002 and 2007, respectively. His research interests include service computing, mobile computing, and business process.

Zibin Zheng is a professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include service computing and cloud computing. He was the recipient of Outstanding PhD Thesis Award of The Chinese University of Hong Kong in 2012, ACM SIGSOFT Distinguished Paper Award at ICSE2010.

Yun Yang received his PhD degree from the University of Queensland, Australia, in 1992, in computer science. He is currently a full professor in the School of Software and Electrical Engineering at Swinburne University of Technology, Melbourne, Australia. His research interests include software technologies, cloud computing, p2p/grid/cloud workflow systems, and service computing.

Supplemental Material

This document contains the supplemental material for the paper "Covering-based Web Service Quality Prediction via Neighborhood-aware Matrix Factorization" submitted to IEEE Transactions on Services Computing.

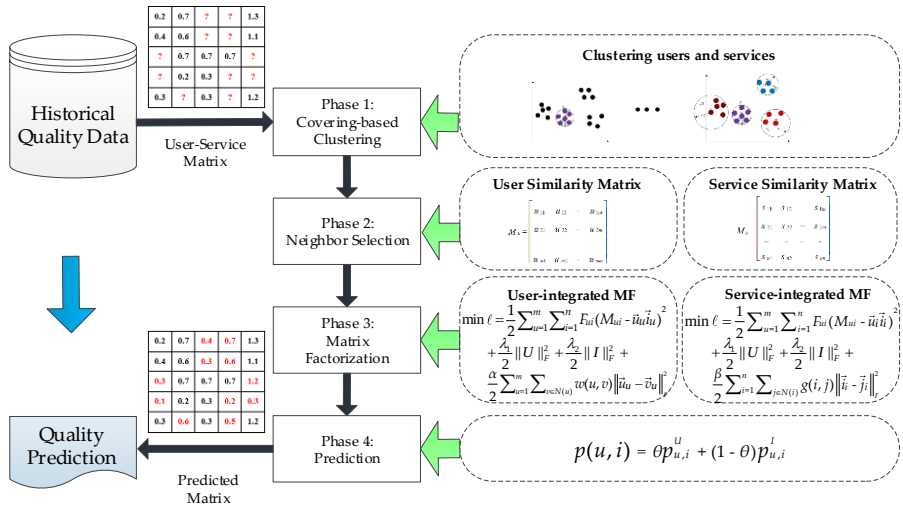


Fig. S1. Overall procedure of CNMF

TABLE S1
NOTATION SUMMARY

Symbol	Description
$N(u)$	User u's neighbors
$N(i)$	Service i's neighbors
U	Set of users
I	Set of Web services
M	User-service matrix
m	Number of users
n	Number of services
U_{CMF}	Latent user feature matrix based on classic MF
I_{CMF}	Latent service feature matrix based on classic MF
$\vec{u}$	Latent user feature vector based on classic MF
$\vec{i}$	Latent service feature vector based on classic MF
d	Dimensionality of the latent feature vector
F_{ui}	Indicator function that returns 1 if user u has invoked Web service i, or 0 otherwise
$U_{CNMF\text{-}U}$	User-based latent user feature matrix based on user-integrated CNMF
$I_{CNMF\text{-}U}$	User-based latent service feature matrix based on user-integrated CNMF
$U_{CNMF\text{-}I}$	Service-based latent user feature matrix based on service-integrated CNMF
$I_{CNMF\text{-}I}$	Service-based latent service feature matrix based on service-integrated CNMF
U_{NIMF}	User-based latent service feature matrix based on user-integrated NIMF
I_{NIMF}	User-based latent service feature matrix based on user-integrated NIMF
M_u	User similarity matrix
M_I	Service similarity matrix
C_{cr}	Current cluster
c_{cr}	Center of the current cluster
r_{cr}	Radius of the current cluster
D_{uc}	Set of data points not yet covered by any cluster
$x_{u,v}$	Number of times that users u and v are assigned into a same cluster
$y_{i,j}$	Number of times that services i and j are assigned into a same cluster
k	Number of neighbors of the user and/or service
$\vec{u}_u$	Latent user feature vector from $U_{CNMF\text{-}U}$
$\vec{u}_i$	Latent user feature vector from $U_{CNMF\text{-}I}$
$\vec{i}_u$	Latent service feature vector from $U_{CNMF\text{-}U}$
$\vec{i}_i$	Latent service feature vector from $I_{CNMF\text{-}I}$
M_{CMF}	User-service matrix based classic MF
M_{NIMF}	User-service matrix based NIMF
$M_{CNMF\text{-}U}$	User-service matrix based CNMF-U

Complexity Analysis

As discussed in Section 4, the procedure of CNMF consists of 4 phases.

In Phase 1, the computational complexity is $O(m)$ since there are a maximum of m data points in D. Similarly, the computational complexity of Phases 2, 3 and 4 are also $O(m)$. Phase 3 and Phase 4 need to be repeated until all data points in D are covered. In Phase 3, the radius of a cluster is the average distance between the center of the cluster and all the data points not covered by any cluster. On average, each newly created cluster covers half of the

data points uncovered. The computational complexity is $O(\log m)$. Thus, 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. Thus, in the worst-case scenario, the computational complexity of clustering m users for all n Web services is $O(mn \log m)$. With a clustering algorithm similar to Algorithm 1, CNMF also partitions the Web services in I into clusters based on their quality values experienced by each of their common users in U. The clustering algorithm is executed for a total of m times, one for each 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, denoted by $y_{i,j}$, $1 \le i, j \le n$, is the total number of times that Web services i and j were partitioned into the same cluster during the m executions of the clustering algorithm. Similar to M_U, M_I is also a symmetric matrix. Similar to Algorithm 1, the computational complexity of the algorithm for clustering Web services is $O(mn \log n)$. Overall, the computational complexity of the entire Phase 1 is $O(mn \log m) + O(mn \log n) = O(mn \log n)$ because there is usually $m \ll n$.

In Phase 2, in order to select the top k users or services, CNMF needs to sort the elements of each entry in M_U and M_I. The computational complexity of the sorting process is dependent on the employed sorting algorithm. Here, we use the computational complexity of the comparison sort algorithm in the worst-case scenario, i.e., $O(m \log m)$, for each entry in M_U and M_I. Overall, the computational complexity of the selection of similar users and similar Web services in the neighbor selection phase is $O(m^2 \log m + n^2 \log n)$.

In Phase 3, the main computational cost occurs during the process of learning latent feature vectors with gradient descent. Take user-integrated MF for example, the computational complexity of the objective function is $O(pd+pkd)$, where p is the number of nonzero quality values in matrix M, k is the number of user u's neighbors, and d is dimensionality of the latent user feature matrix and latent service feature matrix. The computational complexities of gradient $\nabla L' / \nabla \vec{u}_u$ (13) and $\nabla L' / \nabla \vec{i}_u$ (14) are $O(pd+mkd)$ and $O(pd)$ respectively. Usually, there is $k \ll m \ll p$. Therefore, the total computational complexity of one iteration can be expressed by $O(pd)$.

In Phase 4, the prediction based on (20) is simple and runs in O(1).

Thus, the overall computational complexity of CNMF is $O(mn \log m) + O(m^2 \log m + n^2 \log n) + O(pd) + O(1) = O(mn \log m + m^2 \log m + n^2 \log n + pd)$.

Threats to Validity

Threats to construct validity. One of the main threats to the construct validity of our evaluation lies in the comparison of prediction accuracy with the selected prediction methods. The selected prediction methods are based on the CF and MF techniques, which is currently the most popular and most widely employed techniques. There are other methods, e.g., semantics-based prediction methods [26], which are not included in the evaluation. This threat, however, is not significant because CNMF can be indirectly compared with those methods through inspecting the evaluation presented in the relevant literature using the methods included in the evaluation as reference methods. The other main threat to the construct validity of our evaluation is the lack of consideration for aspects like timeliness [27] and locations [28] during the prediction. This threat is also not significant because the consideration of such aspects enhances a prediction method, however does not change the fundamental mechanism. The consideration of timeliness and locations can be included in CNMF in ways similar to [27, 28]. However, a direct comparison between CNMF and other methods in the same category of prediction methods is more straightforward and representative.

Threats to external validity. The main threat to the external validity of our evaluation is the representativeness of the dataset used in the evaluation, which might not be able to exactly represent all real-world applications. To minimize this threat, we used the dataset that has been widely employed in experiments on prediction methods for Web services, i.e., the WS-Dream dataset [18]. In this way, we could compare CNMF with existing methods in a fair and objective manner. In the experiments, we change the matrix density and parameter θ to simulate datasets with different characteristics and inspect their impacts on CNMF. The experimental results demonstrate the effectiveness of CNMF on datasets with similar characteristics. This also significantly reduces the threat to the external validity of our evaluation.

Threats to internal validity. The main threat to the internal validity of our evaluation is its comprehensiveness. Due to the page limit, we have not been able to present the results of experiments under all parameter settings, e.g., more combinations of matrix density and θ values. We believe that this threat is not significant. The exact values, e.g., MAE and RMSE, obtained from the experiments under other parameter settings might be different. However, the advantages of CNMF over the comparing methods are similar to those presented in the paper.

Threats to conclusion validity. The main threat to the conclusion validity of our evaluation is the lack of statistical tests, e.g., chi-square tests. We could have conducted chi-square tests to draw conclusions when evaluating CNMF. However, we ran the experiments for a total of 339 users with 5,828 services and averaged the results each time when we change the parameter setting. This lead to a very large number of test cases, which tend to result in a small p-value in the chi-square tests and lower the practical significance of the test results [29]. However, this number of runs is not even close to the number of observation samples that concern Lin et al. in [29]. Thus, the threat to the conclusion validity due to the lack of statistical tests might be high but not significant.