Predicting Quality of Service for Selection by Neighborhood-Based Collaborative Filtering

Jian Wu, *Member, IEEE*, Liang Chen, *Student Member, IEEE*, Yipeng Feng, Zibin Zheng, *Member, IEEE*, Meng Chu Zhou, *Fellow, IEEE*, and Zhaohui Wu, *Senior Member, IEEE*

***Abstract*—Quality-of-service-based (QoS) service selection is an important issue of service-oriented computing. A common premise of previous research is that the QoS values of services to target users are supposed to be all known. However, many of QoS values are unknown in reality. This paper presents a neighborhood-based collaborative filtering approach to predict such unknown values for QoS-based selection. Compared with existing methods, the proposed method has three new features: 1) the adjusted-cosine-based similarity calculation to remove the impact of different QoS scale; 2) a data smoothing process to improve prediction accuracy; and 3) a similarity fusion approach to handle the *data sparsity* problem. In addition, a two-phase neighbor selection strategy is proposed to improve its scalability. An extensive performance study based on a public data set demonstrates its effectiveness.**

***Index Terms*—Neighborhood-based collaborative filtering (CF), quality-of-service (QoS) prediction, service selection.**

I. Introduction

A SERVICE-ORIENTED COMPUTING (SOC) paradigm and its realization through standardized Web service technologies provide a promising solution to the seamless integration of single-function applications to create new large-grained and value-added services. SOC attracts industry's attention and is applied in many domains, e.g., workflow management, finances, e-business, and e-science. With a growing number of alternative Web services that provide the same functionality but differ in quality properties, the problem of selecting the best performing candidate service is becoming more and more important.

Manuscript received August 5, 2011; accepted April 1, 2012. Date of publication September 12, 2012; date of current version February 12, 2013. This research was supported in part by the National Technology Support Program under Grant 2011BAH15B05, by the National Natural Science Foundation of China under Grant 60873224, by the Zhejiang Provincial Natural Science Foundation of China under Grant Y1110591, by the Science and Technology Program of Zhejiang Province under Grant 2008C03007, and by the National Key Science and Technology Research Program of China under Grant 2009ZX01043-003-003. This paper was recommended by Associate Editor W. Pedrycz.

J. Wu, L. Chen, Y. Feng, and Z. Wu are with the College of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: wujian2000@zju.edu.cn; cliang@zju.edu.cn; birdfeb@zju.edu.cn; wzh@zju.edu.cn).

Z. Zheng is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: zbzheng@cse.cuhk.edu.hk).

M. C. Zhou is with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 200092, China, and also with the New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2012.2210409

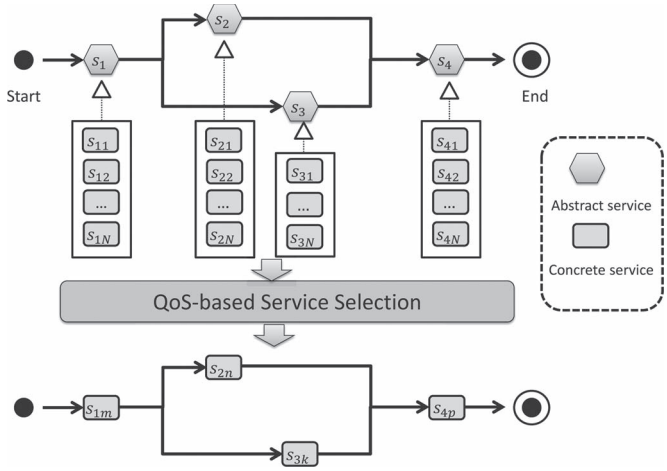


Fig. 1. QoS-based service selection.

Fig. 1 presents a conceptual overview of quality-of-service-based (QoS) service selection. The composite service is composed of several abstract services, and each abstract can be achieved by a set of functionally equivalent concrete services. The goal of QoS-based service selection is to select candidate services from each set of concrete services to satisfy end-to-end QoS requirements and to optimize composite QoS.

There are a number of studies [1]–[5] about QoS-based service selection. Their common premise is that the QoS values of all candidate services to target users are known. However, it may not be true in reality. Due to some factors, e.g., location and network environment, the QoS of the same service to different users may be different. For example, the response time for user u_a (IP:12.108.127.136, USA) to invoke Web service (WSDL: http://biomoby.org/services/wsdl/mmb.pcb.ub.es/parseFeatureAASequenceFromFSOLVText, located in Spain) is 5626 ms, whereas that for user u_b (IP:133.1.74.162, Japan) to invoke the same service is 687 ms [6]. A user can hardly have invoked all services, meaning that the QoS values of services that the user has not invoked are unknown. Hence, providing accurate Web service QoS prediction is very important for service users. Based on the predicted QoS values, desired service selection can be made.

Table I shows a simple example. The numeric value in the pane corresponds to the response time for users to invoke the indicated service. *Null* means that the user has not invoked this service. As the response time of service s_1 to Linda is unknown, we cannot decide which service is the best one for her in terms of response time. To execute QoS-based service selection, a preprocess is necessary to predict the unknown QoS values.

TABLE I
RESPONSE TIME OF SERVICES

	s_1	s_2	s_3
Jason	0.8s	*Null*	*Null*
John	2s	1.2s	0.4s
Linda	*Null*	1.2s	0.4s
Flynn	1s	2s	*Null*

Inspired by the application of neighborhood-based collaborative filtering (CF) [7] in product recommendation, we propose a neighborhood-based CF approach, named ADF, to predict the unknown QoS values. ADF denotes *adjusted* cosine (A-cosine), data smoothing, and *fusion* of similarity results, which are three new features of our prediction approach. As users in a near region or similar network environment appear to have similar QoS of the same service, we use existing QoS data to compute user similarity and predict the unknown QoS values according to the corresponding QoS values of similar users. Service-based prediction can also play an important role in the final result. Different from previous research of service-based prediction, in ADF, we propose to use *A-cosine* equation, which takes the difference in QoS scale between different users into account, to calculate the service similarity, instead of using the *Pearson correlation coefficient (PCC)* approach.

In [8]–[10], unknown QoS of similar neighbors is replaced by zero, which lowers the accuracy of prediction. In ADF, we propose to add a data smoothing process after the selection of similar neighbors and before the prediction of target QoS, to improve the prediction accuracy. In this process, the unknown QoS of similar neighbors is replaced by the average QoS of their cluster. Furthermore, considering the inherent sparsity of QoS data, we propose to use a similarity fusion approach, which makes use of the QoS of similar services to similar users. Therefore, the final QoS is estimated by fusing predictions from three sources: predictions based on QoS of the same service to similar users (similar to user-based CF), predictions based on QoS of similar services to the same user (similar to item-based CF), and those based on QoS of similar services to similar users. This similarity fusion approach is demonstrated to be effective when the density of QoS data is low.

Typically, the scalability problem is the drawback of a memory-based prediction approach. This paper presents a *two-phase neighbor selection (TNS)* strategy to improve the scalability of our proposed approach by accelerating neighbor selection.

In particular, this paper has the following contributions:

1) to improve the accuracy of QoS prediction, we propose a neighborhood-based CF approach called ADF, in which the *A-cosine* approach, the data smoothing process, and the similarity fusion approach are adopted;
2) we propose a TNS strategy to improve the scalability of ADF;
3) we experimentally evaluate ADF by employing a real-world data set.

The rest of this paper is organized as follows. Section II highlights the related work of service selection and QoS prediction. Section III introduces the architecture of our proposed approach. The detailed calculation process is introduced in Section IV. Section V presents our proposed *TNS* strategy, whereas Section VI shows the experimental results. Finally, Section VII concludes this paper.

II. RELATED WORK

The problem of QoS-based service selection has attracted the attention of many researchers and was recently discussed in a number of studies. Ran suggests the use of a "QoS certifier" that certifies the QoS claims made by service providers about their corresponding services [2]. The QoS values may then be incorporated into the "UDDI" registry to facilitate more appropriate service selection. Zeng *et al.* first transfer QoS-based service selection into an optimization problem and present a middleware platform to address the issue of selecting Web services in a way that maximizes user satisfaction measured by QoS attributes while satisfying the constraints set by the user and by the structure of the composite service [3]. Alrifai and Risse first use *mixed integer programming (MIP)* to find the optimal decomposition of global QoS constraints into local constraints and then use distributed local selection to find the best Web services that satisfy these local constraints [1].

A common premise of previous research is that the QoS values of services to target users are all known. However, there are often unknown QoS values to the consumer in reality. Therefore, it is fundamental to predict before any QoS-based service selection.

The CF approach is widely adopted in a recommender system and is often classified as either memory based or model based. In the former, all training data are stored in memory, whereas in the prediction phase, similar objects (users or items) are sorted based on their similarities with the active object. The *cosine* and *PCC* are two widely used methods to compute the similarities between objects. Based on the data from similar users or items, a prediction result can be generated. The most analyzed examples include user-based methods [7], [11], [12], item-based methods [13], [14] and fusion methods [15]. The advantage of the memory-based methods is that they are simple and intuitive on a conceptual level and avoid the complications of a potentially expensive model-building stage. Their drawbacks include the following: 1) the scalability is not good; and 2) nothing is learned from the available user profiles, and little general insight is gained. In the model-based approach, training data are used to generate a prediction model that is able to predict the unknown data. The model examples include decision tree [7], aspect models [16], and latent semantic models [17], [18].

Limited work has been done to predict the unknown QoS values. Shao *et al.* propose a user-based CF algorithm to make similarity mining and predict the QoS of Web services from consumers' experiences [8]. Zheng *et al.* present a hybrid approach called WSRec that combines user-based and item-based approaches together to predict the QoS of Web services, employing two confidence weights to balance these two predicted values [9]. In Section V, we show that ADF outperforms the WSRec approach. Chen *et al.* discover the influence of a user's location to the accuracy of prediction and propose a region-based hybrid CF algorithm to predict the QoS of services [10].

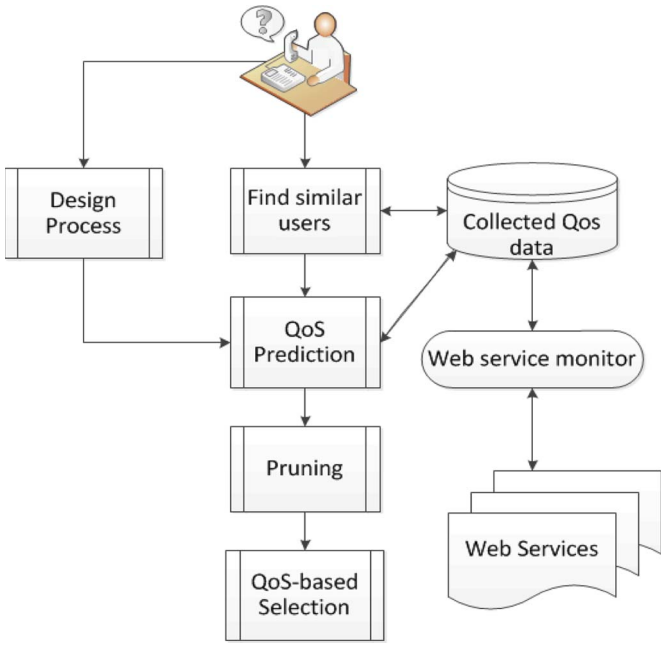


Fig. 2. Architecture of CF-based service selection.

III. Architecture

Fig. 2 shows the architecture of the proposed approach, in which a set of functional-equivalent Web services have been given, and the QoS of Web services is the concern of the requester. When a user's requirement arrives, the procedure of *find similar users* is first employed, whereas the procedure of *design process* should also be employed if this requirement can be only accomplished by composing a set of services. In the procedure of *find similar users*, we utilize the collected QoS data to compute the similarities between the requester and other users for the purpose of finding similar users to the requester. In the procedure of *design process*, a composition process of abstract services is designed to accomplish a user's request. After *find similar users*, we obtain the unknown QoS values of the requester to all candidate Web services by employing the procedure of *QoS Prediction*. Then, the *pruning* procedure is employed to prune the Web services with lower qualities to decrease the number of services for selection, whereas the *QoS-based selection* procedure is employed to select the best candidates through a utility function and to recommend them to the requester.

In this paper, we focus on the research of QoS prediction approach, i.e., the third procedure, leaving other topics as our future work. For the convenience of our readers, Table II summarizes all notations used in this paper.

IV. QoS Prediction

A. Similarity Computation

Given a data set consisting of M service users and N Web services, the invocation records between users and services can be denoted by an $M \times N$ matrix, which is called a user–service matrix. An entry in this matrix $r_{m,n}$ represents a record of invocation (QoS values, e.g., response time and availability). For example, the response time for user m to invoke service n is 100 ms, and availability is 95%. Then, $r_{m,n} = (100, 95\%)$ if we consider these two QoS parameters only. If user m has not invoked service n, then $r_{m,n} =$ null.

Existing work [8]–[10] about QoS prediction uses *PCC* [19] to compute the similarity of users or services. In a user-based CF approach, *PCC* is used to define the similarity between two users u_1 and u_2 based on the services they have commonly invoked using the following:

$$\psi(u_1,u_2)=\frac{\sum_{s\in S}(r_{u_1,s}-\bar{r_{u_1}})^T(r_{u_2,s}-\bar{r_{u_2}})}{\sqrt{\sum_{s\in S}(r_{u_1,s}-\bar{r_{u_1}})^2}\sqrt{\sum_{s\in S}(r_{u_2,s}-\bar{r_{u_2}})^2}} \quad (1)$$

where $S = S_{u_1} \bigcap S_{u_2}$ is the set of services that are both invoked by users u_1 and u_2, $r_{u_1,s}$ is the vector of the QoS values of service s invoked by user u_1, and $\bar{r_{u_1}}$ stands for the vector of average QoS values of the services invoked by user u_1. While using the *PCC* equation, the similarity between users is in the range of $[-1, 1]$ with a larger value indicating that u_1 and u_2 are more similar. Similarly, the detailed *PCC* equation for the similarity computation of services is as follows:

$$\psi(s_1,s_2)=\frac{\sum_{u\in U}(r_{u,s_1}-\bar{r_{s_1}})^T(r_{u,s_2}-\bar{r_{s_2}})}{\sqrt{\sum_{u\in U}(r_{u,s_1}-\bar{r_{s_1}})^2}\sqrt{\sum_{u\in U}(r_{u,s_2}-\bar{r_{s_2}})^2}}. \quad (2)$$

After analyzing real-world Web-service QoS values, we find that the QoS scale of different users is quite different. For example, due to security or gateway, the response time of all services to user A is greater than 3000 ms, whereas that to user B is less than 200 ms due to a fast network. In such situations, the similarity between two services is impacted by other irrelevant issues instead of the service itself when using *PCC*. Considering the differences in QoS scale between different users, we propose to use the *A-cosine* equation [14] to compute the similarity between services as follows:

$$\psi(s_1,s_2)=\frac{\sum_{u\in U}(r_{u,s_1}-\bar{r_u})^T(r_{u,s_2}-\bar{r_u})}{\sqrt{\sum_{u\in U}(r_{u,s_1}-\bar{r_u})^2}\sqrt{\sum_{u\in U}(r_{u,s_2}-\bar{r_u})^2}} \quad (3)$$

where $U = U_{s_1} \bigcap U_{s_2}$ is the set of users who both have invoked services s_1 and s_2. r_{u,s_1} is the vector of QoS values of service s_1 invoked by u, and $\bar{r_u}$ is the vector of average QoS values of service invoked by u. In this *A-cosine* equation, we remove the impact of different QoS scale by using $(r_{u,s_1} - \bar{r_u})$. In ADF, (1) and (3) are employed to compute the user- and service-based similarities, respectively.

B. Similar Neighbor Selection

In Section IV-A, we calculated the similarities between different users, and now, we choose a set of similar neighbors for target users. The process of selecting similar users is crucial for the accuracy of prediction because the prediction of the unknown value depends on the corresponding values of similar users. We find that the QoS values of a user who has very low similarity to the target user are useless or even harmful. Therefore, the traditional *top-K* algorithm is not suitable for this

TABLE II
KEY NOTATIONS AND THEIR DESCRIPTIONS

Notation	Definition and Brief Description
u_i	a user, i=1,2,…,M (M is the total number of users in the collected QoS data)
s_j	a Web service, j=1,2,…,N (N is the total number of services in the collected QoS data)
S_{u_i}	the set of Web services have been invoked by u_i
U_{s_j}	the set of users have invoked s_j
C_{s_j}	the service cluster that s_j belongs to
C_{u_i}	the user cluster that u_i belongs to
ψ	similarity between objects
r_{u_i,s_j}	the QoS of s_j to u_i
$\mathbb{K}$	The number of similar neighbors to be selected in *Hybrid Top-K* algorithm
Ψ	similarity threshold in *Hybrid Top-K* algorithm
Ω	prediction result
$\mathcal{F}$	coefficient weight
T	the number of users in training data
$\mathcal{G}$	the number of Web services invoked by the user in testing data
δ, λ	User-defined weighting factors in the final prediction

scenario. In order to address this problem, we propose a *hybrid top-K* algorithm.

Algorithm 1 Hybrid Top-K Algorithm

Input: u: target user; $T(u)$: set of other users;
Ψ: similarity threshold; and $\mathbb{K}$: the number of similar neighbors to be selected
Output: $S(u)$: set of similar users
1: int $N_{\text{sim}} = 0$
2: **for all** u_i such that $u_i \in T(u)$ and $\psi(u_i, u) \geq \Psi$ **do**
3: $N_{\text{sim}} + +$
4: **end for**
5: **if** $N_{\text{sim}} \geq \mathbb{K}$ **then**
6: $S(u) \leftarrow$ Top-K users
7: **else if** $0 < N_{\text{sim}} < \mathbb{K}$ **then**
8: $S(u) \leftarrow$ Top-N_{sim} users
9: **else if** $\{N_{\text{sim}} = 0$ **then**
10: $S(u) \leftarrow \emptyset$
11: **end if**

In Algorithm 1, we set a similarity threshold Ψ, which is the lower limit of the qualified similar user (lines 2–4). The *top-K* part is executed after the selection of similar neighbors (lines 5–11). Although we remove the effect of dissimilar users by adding Ψ, a second complication is that the number of similar users may be also sometimes small. We propose a similarity fusion approach later to address this issue.

C. QoS *Data Smoothing*

Suppose that u_t is one of the similar users of u and that we want to predict the QoS of service s to user u. Traditional CF-based approaches replace the QoS of s to u_t with 0 if u_t has not invoked s. This lowers the accuracy of predicted QoS. In *ADF*, we add a data smoothing process to improve the accuracy of prediction.

First, we cluster all users based on the user similarity calculated in Section IV-A. There are many algorithms that can be used to perform such clustering. In this paper, a *K-means* algorithm [20] is selected. Due to limited space, we do not introduce it in detail. Assuming that all users could be clustered into k groups, clustering results of the users $U = \{u_1, u_2, \ldots, u_n\}$ are represented as $\{C_u^1, C_u^2, \ldots, C_u^k\}$.

Given u_t that belongs to cluster C_u, $C_u \in \{C_u^1, C_u^2, \ldots, C_u^k\}$, then the equation to calculate QoS vector $r_{u_t,s}$ is as follows:

$$r_{u_t,s} = \bar{r_{u_t}} + \Delta r_{C_u}(s) \tag{4}$$

where $\Delta r_{C_u}(s)$ is the average QoS deviations of service s to all users in cluster C_u

$$\Delta r_{C_u}(s) = \frac{\sum_{u' \in C_u(s)} (r_{u',s} - \bar{r_{u'}})}{|C_u(s)|} \tag{5}$$

where $C_u(s) \in C_u$ is the set of the users in cluster C_u who have invoked service s, and $|C_u(s)|$ is the cardinality of $C_u(s)$.

D. Prediction

User-based prediction uses the data of similar users to predict the unknown value of target service s to target user u as follows:

$$\Omega_u = \bar{r_u} + \frac{\sum_{u_1 \in S(u)} \psi(u_1, u)(r_{u_1,s} - \bar{r_{u_1}})}{\sum_{u_1 \in S(u)} \psi(u_1, u)} \tag{6}$$

where $\bar{r_u}$ is the vector of average QoS values of services invoked by u, $S(u)$ is the set of u's similar users, and $\bar{r_{u_1}}$ is the vector of average QoS values of services invoked by u_1. Service-based prediction is given as follows:

$$\Omega_s = \bar{r_s} + \frac{\sum_{s_1 \in S(s)} \psi(s_1, s)(r_{u,s_1} - \bar{r_{s_1}})}{\sum_{s_1 \in S(s)} \psi(s_1, s)} \tag{7}$$

where $\bar{r_s}$ is the vector of average QoS values of s invoked by different users, and $\bar{r_{s_1}}$ is the vector of average QoS values of s_1 invoked by different users.

However, the methods [8]–[10] ignore the information that can be obtained from the QoS of similar services to other but similar users. Not using such QoS information causes the data sparsity problem: For many unknown QoS values, no reliable prediction can be made because of a lack of similar users and services. To solve it, we propose to make use of the QoS information of similar services to similar users and integrate them into the final prediction result. Therefore, the final QoS is estimated by fusing predictions from three sources: predictions based on QoS of the same service to similar users, those based

on QoS of similar services to the same user, and those based on QoS of similar services to similar users.

Suppose that u_1 is the similar user of u and s_1 is the similar service of s, then the similarity between r_{u_1,s_1} and $r_{u,s}$ is:

$$\psi(u_1 s_1, us) = \psi(u_1, u) \times \psi(s_1, s). \tag{8}$$

The user-and-service-based prediction uses the QoS of similar services to similar users to predict the target unknown QoS as follows:

$$\Omega_{us} = \frac{\sum_{u_1 \in S(u), s_1 \in S(s)} \psi(u_1 s_1, us) P_{u,s}(u_1, s_1)}{\sum_{u_1 \in S(u), s_1 \in S(s)} \psi(u_1 s_1, us)} \tag{9}$$

where $P_{u,s}(u_1, s_1)$ is the prediction of $r_{u,s}$ made by r_{u_1,s_1}, and $P_{u,s}(u_1, s_1)$ is computed as follows:

$$P_{u,s}(u_1, s_1) = \left(r_{u_1,s_1} - \frac{\bar{r_{u_1}} + \bar{r_{s_1}} - \bar{r_u} - \bar{r_s}}{2} \right) \tag{10}$$

where $\bar{r_{u_1}}$ and $\bar{r_u}$ are the vectors of average QoS values of services invoked by u_1 and u, respectively, and $\bar{r_{s_1}}$ and $\bar{r_s}$ are the vectors of average QoS values of s_1 and s, respectively.

Since these three prediction results are all computed based on similarities, we should compute the confidence weight of each prediction result. The confidence weight of user-based prediction is defined as follows:

$$\mathcal{F}_u = \sum_{u_1 \in S(u)} \frac{\psi(u_1, u)}{\sum_{u_1 \in S(u)} \psi(u_1, u)} \times \psi(u_1, u). \tag{11}$$

The confidence weight of service-based prediction is

$$\mathcal{F}_s = \sum_{s_1 \in S(s)} \frac{\psi(s_1, s)}{\sum_{s_1 \in S(s)} \psi(s_1, s)} \times \psi(s_1, s). \tag{12}$$

The confidence weight of user-and-service-based prediction is

$$\mathcal{F}_{us} = \sum_{u_1 \in S(u), s_1 \in S(s)} \frac{\psi(u_1 s_1, us)^2}{\sum_{u_1 \in S(u), s_1 \in S(s)} \psi(u_1 s_1, us)}. \tag{13}$$

The value of confidence weight is in the range of [0,1] with a larger value indicating that the corresponding result is more preferable. As the final prediction result is the aggregation of all the prediction values, we set two parameters λ and $\delta \in [0, 1]$ to determine how it relies on each individual prediction. When $S(u) \neq \emptyset$ and $S(s) \neq \emptyset$, the final equation for QoS prediction is

$$\Omega = w_u \times \Omega_u + w_s \times \Omega_s + w_{us} \times \Omega_{us} \tag{14}$$

where w_{us}, w_u, and w_s stand for the participation that each predicted result takes in the final prediction, and $w_{us} + w_u + w_s = 1$. They are computed as follows:

$$\begin{aligned} w_{us} &= \frac{\delta \mathcal{F}_{us}}{\delta \mathcal{F}_{us} + \lambda(1-\delta)\mathcal{F}_u + (1-\lambda)(1-\delta)\mathcal{F}_s} \\ w_u &= \frac{\lambda(1-\delta)\mathcal{F}_u}{\delta \mathcal{F}_{us} + \lambda(1-\delta)\mathcal{F}_u + (1-\lambda)(1-\delta)\mathcal{F}_s} \\ w_s &= \frac{(1-\lambda)(1-\delta)\mathcal{F}_s}{\delta \mathcal{F}_{us} + \lambda(1-\delta)\mathcal{F}_u + (1-\lambda)(1-\delta)\mathcal{F}_s}. \end{aligned} \tag{15}$$

The parameters δ and λ mean the participation that the corresponding prediction result takes in the final result. If $0 < \delta < 1$ and $0 < \lambda < 1$, ADF aggregates the final results from three sources. If $\delta = 0$ and $0 < \lambda < 1$, ADF turns to be a hybrid approach aggregating the user-based and service-based prediction results only. Similarly, if $\lambda = 0$ and $0 < \delta < 1$, it combines the service-based and user-and-service-based results together. There are four special cases: 1) $\delta = 0$ and $\lambda = 1$, $\Omega = \Omega_u$; 2) $\delta = 0$ and $\lambda = 0$, $\Omega = \Omega_s$; 3) $\delta = 1$ and $\lambda = 0$, $\Omega = \Omega_{us}$; and 4) $\delta = 1$ and $\lambda = 1$, $\Omega = \Omega_{us}$.

V. TNS

Suppose that Mike is a new user to this system and has invoked some Web services. The traditional approaches [8], [9] to predict the QoS values of other Web services to Mike proceed with: 1) calculating the similarity between Mike and all users who have been in the database; 2) selecting neighbors according to their similarity to him; and 3) predicting the QoS values of other Web services to him according to the records of his neighbors. Such approaches suffer from poor scalability when more and more new users and new services are added into the recommender system. That is one main disadvantage of memory-based CF approach.

Next, we propose the TNS strategy to improve the scalability by accelerating steps 1 and 2.

A. Neighbor Preselection

In the process of *QoS data smoothing*, we have clustered all users based on the user similarities. In this phase, we utilize these user clusters to realize neighbor preselection. The feature of users in the same cluster can be represented by the centroid of the cluster, whereas the centroid can be represented as average QoS over all users in the cluster. Therefore, the similarity between cluster C_u and target user u_1 can be calculated as follows:

$$\psi(u_1, C_u) = \frac{\sum_{s \in S(t)} \Delta r_{C_u}(s)(r_{u_1,s} - \bar{r_{u_1}})}{\sqrt{\sum_{s \in S(t)} (\Delta r_{C_u}(s))^2} \sqrt{\sum_{s \in S(t)} (r_{u_1,s} - \bar{r_{u_1}})^2}} \tag{16}$$

where $S(t) = S_{u_1} \bigcap S_{C_u}$ means the set of services that have been both invoked by u_1 and some in C_u, and $\Delta r_{C_u}(s)$ is given by (5). In particular, the similarity between u_1 and C_u are the same as the similarity between u_1 and the centroid of C_u. After calculating the similarity between each cluster and the target user, we can get $C_u^x \in \{C_u^1, C_u^2, \ldots, C_u^k\}$, i.e., the most similar cluster to u_1. As the users in cluster C_u^x is clustered according to the similarity between each other, we take the users in C_u^x as the candidates for neighbor selection.

As for the neighbor preselection for a target service, we can also use this cluster-based approach since the process of clustering the existing services according to similarity can be executed offline. Similarly, given a target service s_1 and a set

of clusters $\{C_s^1, C_s^2, \ldots, C_s^k\}$, the similarity between s_1 and cluster C_s can be calculated as follows:

$$\psi(s_1, C_s) = \frac{\sum_{u \in U(t)} \Delta r_{C_s}(u)(r_{u,s_1} - \bar{r_u})}{\sqrt{\sum_{u \in U(t)} (\Delta r_{C_s}(u))^2} \sqrt{\sum_{u \in U(t)} (r_{u,s_1} - \bar{r_u})^2}} \tag{17}$$

where $U(t) = U_{s_1} \bigcap U_{C_s}$ means the set of users who have invoked s_1 and some in C_s, and $\Delta r_{C_s}(u)$ is computed as follows:

$$\Delta r_{C_s}(u) = \frac{\sum_{s' \in C_s(u)} (r_{u,s'} - \bar{r_u})}{|C_s(u)|} \tag{18}$$

where $C_s(u) \in C_s$ is the set of services in cluster C_s that have been invoked by user u. Similarly, we can obtain the most similar service cluster $C_s^y \in \{C_s^1, C_s^2, \ldots, C_s^k\}$ to s_1 and take the services in C_s^y as the candidates for neighbor selection.

By employing the process of neighbor preselection, we improve the efficiency of neighbor selection, as only users/services in the similar clusters are considered for neighbor candidates.

B. Neighbor Selection

After obtaining the candidate set from the preselection process, we select neighbors for the target user (service) in two steps: 1) recalculating similarity between a user (service) in the candidate set and the target one; and 2) selecting neighbors according to the similarity. For step 1, we use (1) [(3) for service], and for step 2, the *hybrid top-K* algorithm is employed.

C. Analysis

As the similarity calculation is the most time-consuming operation, we compare the efficiency between the traditional approach and proposed *TNS* in terms of the number of times needed to compute similarity. In the former case, the step of "*calculating the similarity between Mike and all users who have been in the database*" costs N (the number of all users) similarity calculations, whereas the next step of "*selecting neighbors according similarity*" also costs N calculations.

In *TNS*, as the clustering process is executed offline, only K (average number of clusters) similarity calculations are needed in the first phase. In the second phase, it needs N/K. Therefore, the number of similarity calculations in *TNS* is $K + N/K$, which achieves its minimum value $2\sqrt{N}$ when $K = \sqrt{N}$. Compared with the traditional approach ($2N$ calculations), *TNS* is more efficient.

Moreover, the maintenance cost of *TNS* is low. When a new user (service) is added, we first use *TNS* and *ADF* to predict the unknown QoS values and then add this user (service) into the cluster that is the most similar with the user (service). Note that recalculating the centroid of the cluster is needed.

VI. Experiments

In this section, we present an experimental evaluation of the proposed approach by measuring the following: 1) performance of different approaches, in terms of *normalized mean absolute error (NMAE)*; 2) impact of A-cosine, data smoothing and similarity fusion to the performance of *ADF*; 3) impact of parameter $\mathbb{K}$, Ψ, λ, and δ to the performance of *ADF*; and 4) performance of *TNS*.

A. Experimental Setup

We have conducted our experiments using a public real-world Web-service QoS data set, which is collected in [9]. It contains the records of 1 974 675 Web service invocations executed by 339 distributed service users on 5825 Web services. The record of each invocation contains two parameters: *response time* and *throughput*. More details about this data set can be found in [6]. In this paper, we randomly extract 150 users and 100 Web services and use the invocation records between them as the experimental data.

Our experiments are implemented with JDK1.6.0_21, Eclipse 3.6,0, and mySQL 5.0. They are conducted on a Dell Inspire 13R machine with 2.27-GHz Intel Core I5 CPU and 2GB RAM, running a Windows 7 operating system.

B. Evaluation Metric

In our experiments, NMAE is used to evaluate the accuracy of prediction. The mean absolute error (MAE) is as follows:

$$\text{MAE} = \frac{\sum_{U,S} |r_{u,s} - \hat{r_{u,s}}|}{N} \tag{19}$$

where $r_{u,s}$ is the predicted QoS value of service s observed by user u, $\hat{r_{u,s}}$ stands for the expected or real QoS value, and N is the total number of predictions. As we know, the QoS value range of services may differ so tremendously that the use of only MAE is not objective enough. As an adjustment, NMAE normalizes the differences range of MAE by computing

$$\text{NMAE} = \frac{\text{MAE}}{\sum_{U,S} \frac{r_{u,s}}{N}}. \tag{20}$$

The smaller NMAE is, the more accurate QoS prediction will be.

C. Performance Comparison

We compare the proposed approach with three representative prediction methods: a user-based algorithm using PCC (UPCC) [7], an item-based algorithm using PCC (IPCC) [14], and WSRec [9]. WSRec combines the UPCC and IPCC results together and set a parameter to balance the value of each other. Note that (1) and (2) are employed to calculate UPCC and IPCC, respectively. To the best of our knowledge, the WSRec approach is the best one for QoS prediction at present.

As the data set used for experiments is the set of invocation records between 150 users and 100 Web services, we create a 150 $\times$ 100 user–service matrix, where each entry in it is a vector including two QoS values: *response time* and *throughput*. During the experiment, the 150 $\times$ 100 matrix is divided into two

TABLE III
COMPARISON OF PREDICTION ACCURACY (A SMALLER VALUE MEANS BETTER PERFORMANCE)

Density	Methods	T = 100						T =140					
		Response Time			Throughput			Response Time			Throughput		
		$\mathcal{G}5$	$\mathcal{G}10$	$\mathcal{G}20$	$\mathcal{G}5$	$\mathcal{G}10$	$\mathcal{G}20$	$\mathcal{G}5$	$\mathcal{G}10$	$\mathcal{G}20$	$\mathcal{G}5$	$\mathcal{G}10$	$\mathcal{G}20$
5%	UPCC	0.7957	0.7791	0.7567	0.7836	0.7246	0.6822	0.7703	0.7336	0.7148	0.7267	0.6933	0.6578
	IPCC	0.6767	0.6540	0.6105	0.5986	0.5820	0.5281	0.6547	0.5975	0.5464	0.5524	0.5377	0.5108
	WSRec	0.6093	0.5744	0.5428	0.6093	0.5744	0.5428	0.5548	0.5294	0.5049	0.5499	0.5208	0.4989
	ADF	**0.5859**	**0.544**	**0.5012**	**0.5677**	**0.5412**	**0.5139**	**0.5318**	**0.5166**	**0.4857**	**0.5225**	**0.4998**	**0.477**
10%	UPCC	0.747	0.7291	0.7074	0.722	0.6675	0.6264	0.6996	0.6572	0.6336	0.656	0.6203	0.6043
	IPCC	0.5916	0.5593	0.5289	0.5602	0.5388	0.476	0.5411	0.5022	0.4603	0.5264	0.482	0.4664
	WSRec	0.548	0.4984	0.4728	0.5317	0.5106	0.4704	0.4918	0.4624	0.457	0.5153	0.4708	0.4506
	ADF	**0.5208**	**0.4874**	**0.4592**	**0.521**	**0.5091**	**0.4637**	**0.4789**	**0.455**	**0.4425**	**0.4901**	**0.4674**	**0.4327**
15%	UPCC	0.7213	0.7070	0.6802	0.6821	0.6298	0.5949	0.6758	0.6369	0.6253	0.6285	0.6086	0.5664
	IPCC	0.5566	0.4904	0.4623	0.5179	0.4762	0.4583	0.5184	0.4448	0.4223	0.4982	0.4478	0.4284
	WSRec	0.4854	0.4627	0.4413	0.5089	0.4698	0.4458	0.4709	0.4299	0.4116	0.4825	0.4403	0.4185
	ADF	**0.4767**	**0.4429**	**0.4256**	**0.485**	**0.4632**	**0.4341**	**0.4698**	**0.4208**	**0.4056**	**0.4775**	**0.4387**	**0.4063**

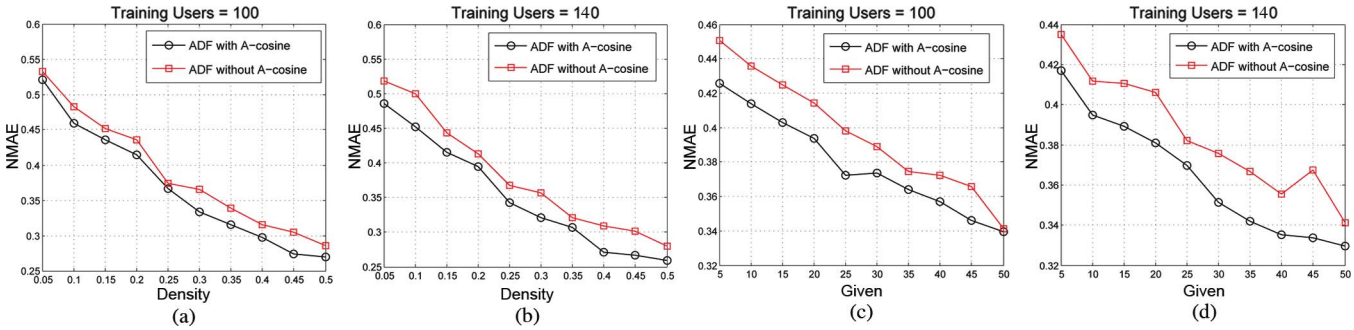


Fig. 3. Impact of A-cosine (Note that the word *given* in below figures means $\mathcal{G}$).

parts: N rows as the training matrix and the other $(150 - N)$ rows as the testing matrix. The users in the testing matrix are called as target users. Then, the training matrix density is thinned randomly to $m\%$ to simulate the situation in which one user in the training matrix has employed only $m\%$ of all services. This step is used to make the condition of experiments similar to that of a real scenario. In addition, we vary the number of invocation records that target users can provide, i.e., the number of Web services that they have invoked. To minimize error, each experiment is looped 50 times, and the average value is reported.

Table III shows the prediction performance of above four approaches on *response time* and *throughput* employing 5%, 10%, and 15% density of the training matrix, respectively. For the users in testing matrix (target users), we vary the number of invoked Web services as 5, 10, and 20 by randomly sampling (named as $\mathcal{G}5$, $\mathcal{G}10$, $\mathcal{G}20$ in Table III). In addition, we consider the influence of the size of the training matrix and vary the number of training users, i.e., $T = 100$ or 140. Empirically, we set $\lambda = 0.1$, $\delta = 0.05$, $\Psi = 0.25$, and $\mathbb{K} = 10$ (the number of similar neighbors in Algorithm 1). From Table III, we find that *ADF* obtains smaller NMAE values, which means higher prediction accuracy in all cases. This demonstrates that ADF gives more accurate prediction. Comparing the prediction results for cases of $T = 100$ and 140, we can find that the latter's NMAE values are smaller, which indicates that the increase in T improves the prediction accuracy. Similarly, the increase in the density of a training matrix improves prediction accuracy as well since higher density means more training data. Furthermore, the increase in the number of invoked services ($\mathcal{G}5$, $\mathcal{G}10$, and $\mathcal{G}20$) also improves it.

D. Evaluation of A-cosine

One of the differences between *ADF* and the traditional PCC-based approach is the *A-cosine* equation used in *ADF*. To study its impact to the performance of *ADF*, we implement two versions of *ADF*: one version employs PCC and *A-cosine* for the similarity calculation, i.e., (1) and (3), whereas the other one only employs (1) and (2). In the experiment, we set $\lambda = 0.1$, $\delta = 0.05$, $\Psi = 0.25$, and $\mathbb{K} = 10$.

In Fig. 3(a) and (b), $\mathcal{G}$ is fixed as 10. In Fig. 3(a), we can find that ADF with *A-cosine* outperforms the ADF without it with the density of a training matrix varying from 5% to 50%. In particular, the influence of *A-cosine* is obvious when the density is from 25% to 50%. Fig. 3(b) shows the performance comparison when $T = 140$, and the result is similar to Fig. 3(a).

In Fig. 3(c) and (d), the density of the training matrix is fixed as 20%. In Fig. 3(c), we can find that ADF with *A-cosine* outperforms ADF without it with $\mathcal{G}$ varying from 5 to 50. When the number of T is 140, the improvement that *A-cosine* brings to ADF is clearer.

Fig. 3 shows that *ADF* with *A-cosine* obtains better prediction accuracy. This is because the PCC equation for item similarity calculation does not consider that the QoS scales of different users are quite different, which lowers the accuracy of prediction. Clearly, *A-cosine* equation's usage enhances the

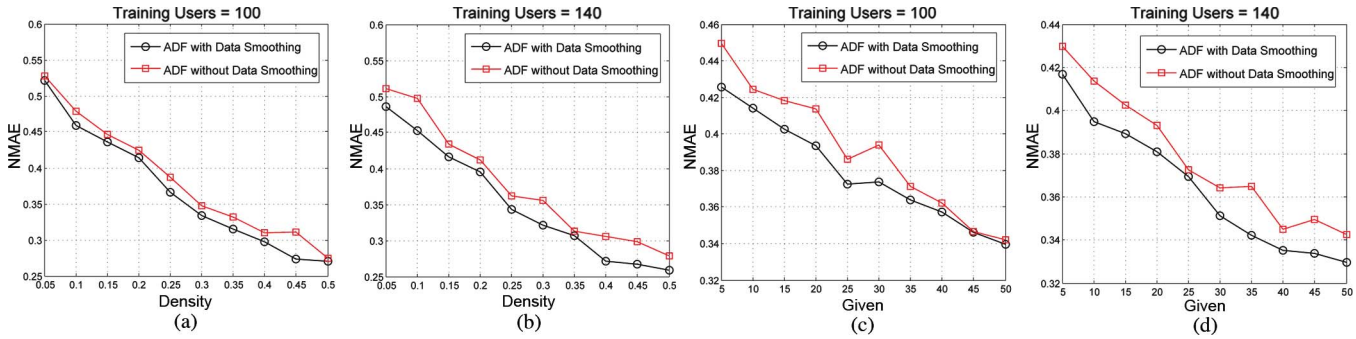


Fig. 4. Impact of data smoothing.

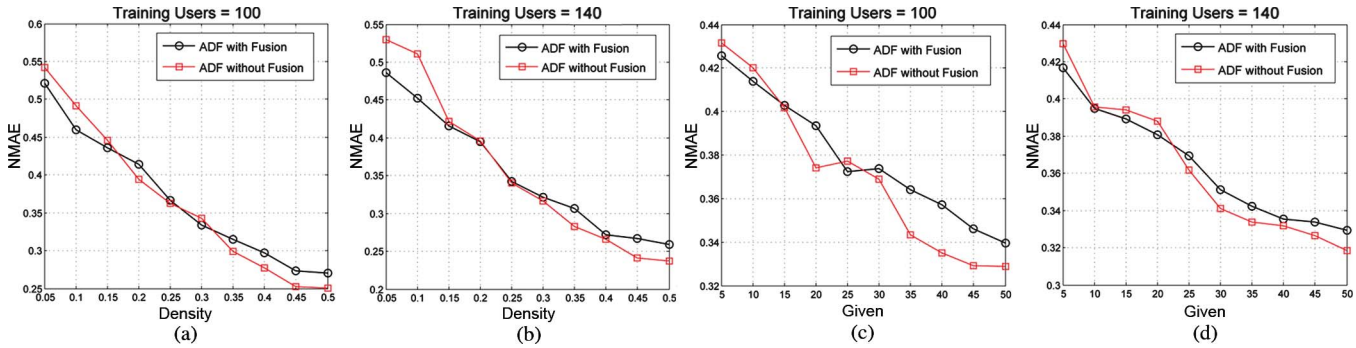


Fig. 5. Impact of fusion.

accuracy of prediction by removing the impact of different QoS scales.

E. Evaluation of Data Smoothing

A data smoothing process is one of main contributions in this paper. In order to study its impact, we implement two versions of *ADF*: one with the proposed data smoothing process, i.e., (4) and (5), and the other without it.

In Fig. 4(a) and (b), we find ADF with data *smoothing* outperforms ADF without it when the density of the training matrix varies from 5% to 50%. In particular, the gap between their performances is bigger when the number of T is 140. Fig. 4(c) and (d) also demonstrates that the process of data smoothing can improve the accuracy of prediction when $\mathcal{G}$ varies.

In Fig. 4, we can find that the data smoothing process improves the accuracy of QoS prediction. This is because the traditional methods replace the QoS of service s to u_t with 0 if u_t has not invoked service s. This kind of action lowers the prediction accuracy. A data smoothing process replaces this unknown QoS with an appropriate value by using a cluster algorithm.

F. Evaluation of Similarity Fusion

As previously said, the process of similarity fusion is used to handle the data *sparsity* problem. To study its impact, we implement two versions of *ADF*: one with a similarity fusion process and the other without it. In the experiment, we set density $= 2\%$, $\lambda = 0.1$, $\delta = 0.05$, $\Psi = 0.25$, and $\mathbb{K} = 10$.

Different from the results in Figs. 3 and 4, the performance of ADF with a similarity fusion process is not always better than the ADF without it in Fig. 5. In Fig. 5(a), the NMAE value of ADF with a similarity fusion process is smaller when the density of a training matrix is smaller than 15%, whereas the performance of ADF without it is better when the density is over 15%. In Fig. 5(b), ADF with a similarity fusion process is more accurate when the density is smaller than 25%, whereas the ADF without it is more accurate when the density is over 25%.

In Fig. 5(c) and (d), we fix the density of the training matrix and vary the value of $\mathcal{G}$. Similar with Fig. 5(a) and (b), the performance of ADF is better when the value of $\mathcal{G}$ is below a threshold, whereas the performance of ADF without it is better when the value of $\mathcal{G}$ is over the threshold.

Fig. 5 demonstrates that the similarity fusion process is favorable when the density of a training matrix or $\mathcal{G}$ is lower than a threshold, and is not useful or even harmful when the data density exceeds the threshold. Therefore, we must first estimate the distribution of experimental data before deciding whether to use it.

G. Impact of the Hybrid Top-K Algorithm

The selection of similar users or items is very important in the CF-based QoS prediction approach. In this paper, we propose a *hybrid top-K* algorithm to select neighbors. To study its performance, we set $T = 100$, density $= 20\%$, $\lambda = 0.1$, and $\delta = 0.05$. In Fig. 6, we can observe that the performance of $\mathbb{K} = 15$ outperforms the other two ($\mathbb{K} = 10$ and $\mathbb{K} = 20$). Therefore, we see that more candidates do not contribute to higher accuracy. As previously discussed, a dissimilar candidate is not useful and, sometimes, detrimental to accuracy.

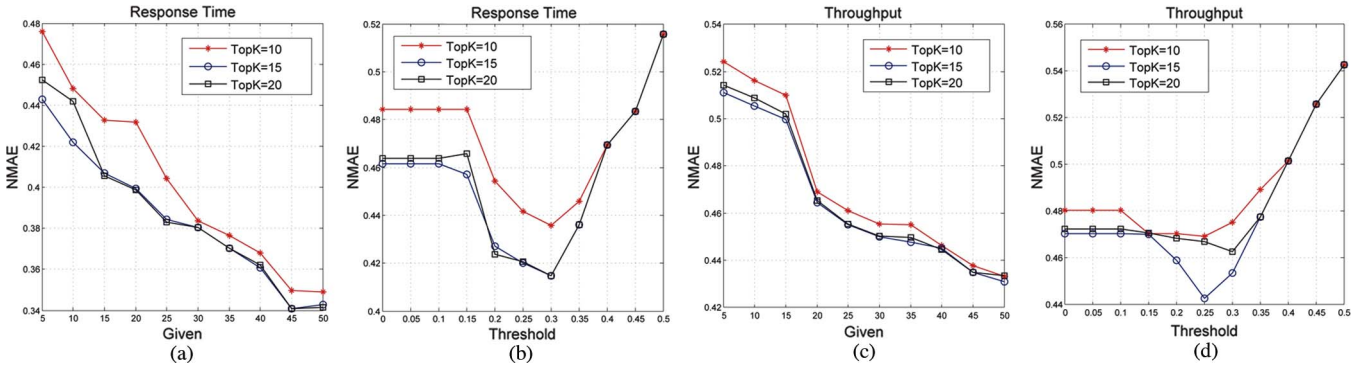


Fig. 6. Impact of hybrid top-K algorithm where *threshold* means Ψ.

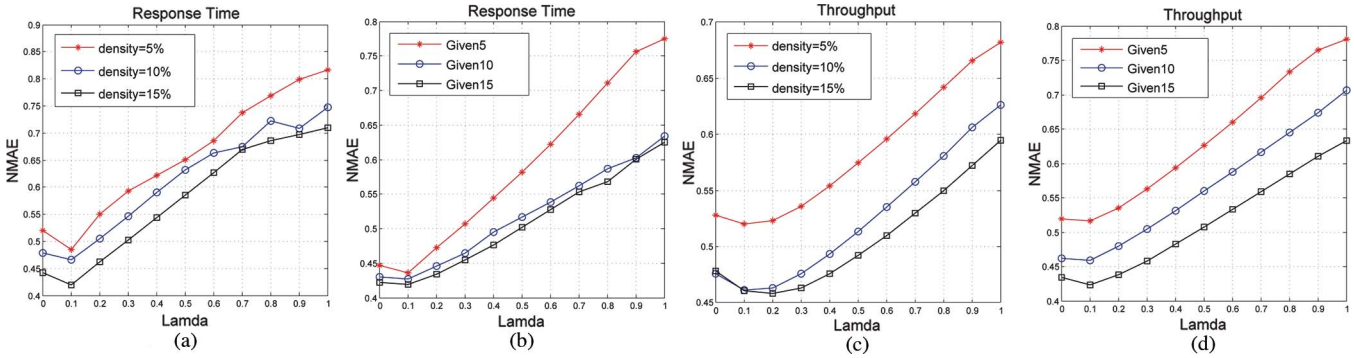


Fig. 7. Impact of λ.

In Fig. 6(b)–(d), we fix $\mathcal{G}$ and vary Ψ to study the relation between Ψ and performance of *ADF*. In Fig. 6(b), we can clearly observe that NMAE first decreases and then increases with Ψ. This is because when similarity threshold Ψ is small, some objects with low similarity values are selected as neighbors, which results in a more imprecise prediction. Similarly, when Ψ is large, only a few objects or even no object can be selected as neighbors, which renders the result inaccurate. Therefore, we can draw the conclusion that Ψ impacts the prediction results, and a more suitable Ψ can provide more accurate prediction results. Based on the observations, the optimal value of Ψ in Fig. 6(b) is 0.3, whereas in Fig. 6(d), it shifts from 0.25 to 0.3. Thus, the optimal Ψ is not fixed but influenced by the nature of data sets.

H. Impact of λ

In *ADF*, the final predicted value depends on three parts: user-based, service-based and user-and-service-based predictions. Parameters λ and δ adjust the weight of the three parts in macro presentation, although it takes a more complex computation in (15).

To study the impact of λ, we set $T = 100$, $\Psi = 0.25$, and $\mathbb{K} = 10$. In this evaluation, the value of δ is initially set as 0.05, and the value of λ varies from 0 to 1. Fig. 7(a) and (c) show the trend of *response time* and *throughput* with the density of the training matrix varies from 5% to 15%. Fig. 7(b) and (d) shows the results of $\mathcal{G} = 5$, 10, and 20 with a 20% density training matrix of *response time* and *throughput*, respectively. When $\lambda = 0$, the weight of user-based prediction is 0, the weight of service-based prediction is $w_s = (1-\delta)\mathcal{F}_s/\delta\mathcal{F}_{us} + (1-\delta)\mathcal{F}_s$, and the weight of user-and-service-based prediction is $w_{us} = \delta\mathcal{F}_{us}/\delta\mathcal{F}_{us} + (1-\delta)\mathcal{F}_s$. As the value of δ is set as 0.05, the result of ADF is almost that of a service-based prediction approach. Similarly, the result of ADF is almost that of a user-based prediction approach when $\lambda = 1$. In Fig. 7, we can observe that the service-based prediction is much more accurate than the user-based prediction as the NMAE value of $\lambda = 0$ is greatly smaller than that of $\lambda = 1$. Therefore, we should put a high weight on service-based prediction. In addition, we can draw conclusion that λ impacts the prediction results significantly, and a more suitable λ can also provide more accurate prediction result.

In Fig. 7(a), as the density of the training matrix varies from 5% to 15%, the best prediction performance is reached when $\lambda = 0.1$. However, in Fig. 7(c), the optimal λ value shifts from 0.1 to 0.3 as the density of the matrix varies from 5% to 15%. The difference indicates that the optimal λ value is not only influenced by the density of training data but also by the nature of data sets. In Fig. 7(b) and (d), we fix the density of the training matrix and vary the value of $\mathcal{G}$. In Fig. 7(b) and (d), the optimal λ value is 0.1. This indicates that it is not influenced by $\mathcal{G}$.

I. Impact of δ

The value of δ stands for the weight of user-and-service-based prediction. When $\delta = 0$, the predicted value of *ADF* depends on user-based and service-based prediction only. To study its impact, we set $T = 100$, $\Psi = 0.25$, and $\mathbb{K} = 10$. In this experiment, λ is set as 0.1 according to the conclusion in the previous section, and δ varies from 0 to 1. Fig. 8(a) and (c)

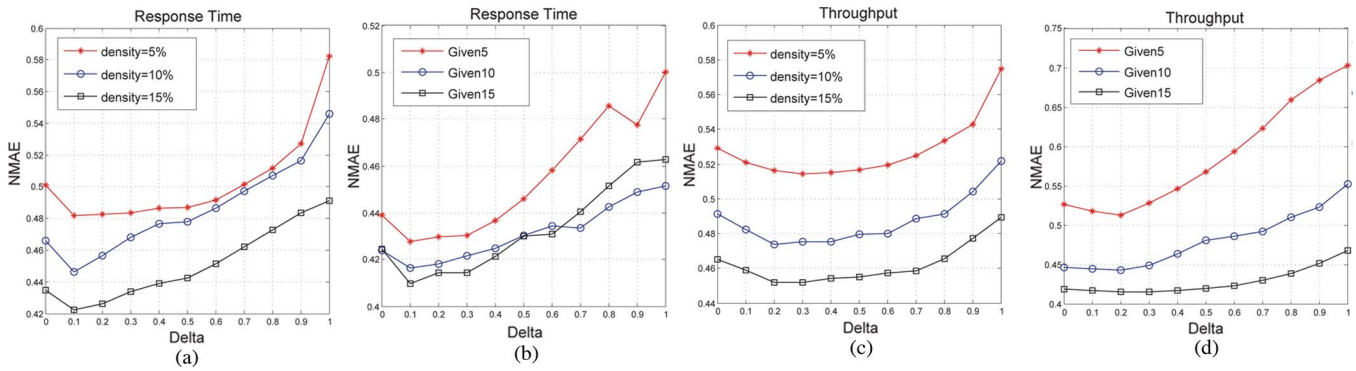


Fig. 8. Impact of δ.

shows *response time* and *throughput* when density $= 5\%$, 10%, and 15% with $\mathcal{G} = 20$. Fig. 8(b) and (d) shows *response time* and *throughput* when $\mathcal{G} = 5$, 10, and 20 with 20% density of the training matrix.

In Fig. 8, we can observe that an optimal δ value improves the accuracy of the prediction, whereas an unsuitable δ value makes it worse. In Fig. 8(c), the optimal δ value shifts from 0 to 0.2 with the density varies from 5% to 15%. This indicates that it is influenced by the density of training data. In Fig. 8(b), the optimal δ value is 0.1 as $\mathcal{G}$ varies from 5 to 15. In Fig. 8(d), it is 0.2. Therefore, it is not influenced by $\mathcal{G}$ but by the nature of data sets for this example.

Through evaluating the impact of λ and δ, we find that both data density and the nature of data sets should be considered in the process of selecting optimal λ and δ values.

J. Performance of TNS strategy

As discussed earlier, a memory-based CF approach suffers from poor scalability because the similarities between new user (service) and all existing users (services) must be computed once a new user (service) is added. Neighbors are then selected based on these similarities, and unknown QoS values are predicted based on those of neighbors. Because the similarity calculation, which is used for neighbor selection, costs much time, *TNS* is proposed to handle this problem.

To evaluate the performance of *TNS*, we implement two versions of *ADF*: one version with *TNS*, and the other without it. In the experiment, we set density $= 20\%$, $\mathbb{K} = 10$, $\Psi = 0.35$, $T = 140$, $\mathcal{G}$ is 10, $\lambda = 0.1$, and $\delta = 0.05$. As the training matrix size is 140, the matrix size for prediction is 10. The ten vectors (contains some services' QoS to one user) are inserted into the prediction system one by one. Once a new vector is inserted, we predict the unknown QoS values for the user that this vector belongs to, and record the computation time t_i. *Computation time* in Fig. 9 refers to the sum of all t_i.

In Fig. 9, we can find that the introduction of *TNS* in ADF greatly improves the efficiency of QoS prediction. When #Cluster(the number of clusters) $= 10$, the computation time of *ADF* with *TNS* is only about 60% of that of the *ADF* without it. It can be easily explained by the complexity analysis in Section V-C. In particular, the relation between computation time and $\#Clusters$ is not monotonic. When the latter increases from 0 to 12, the computation time decreases. It

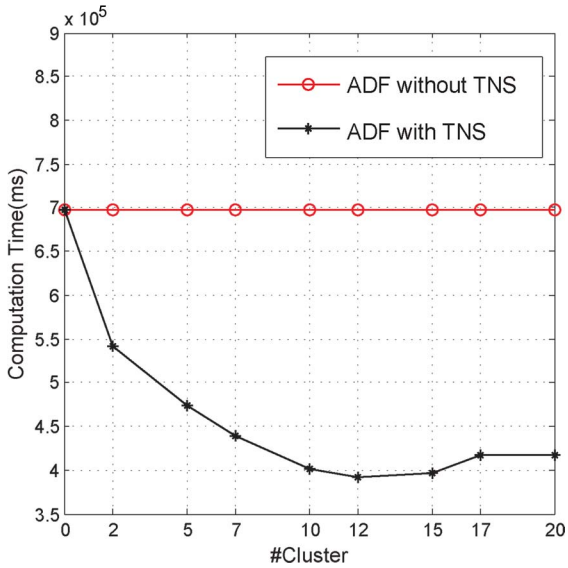


Fig. 9. Impact of TNS strategy.

increases when #Clusters increases from 12 to 20. In our analysis about *TNS* strategy in Section V-C, the time achieves its minimum value when $K = \sqrt{N}$. In this case, $N = 150$, and $\sqrt{N} \approx 12$. The experimental outcome validates our analysis result.

VII. Conclusion

In this paper, we have presented a neighborhood-based CF approach called *ADF* to predict the unknown QoS values. Different from the previous methods, we have use *A-cosine* equation to compute the service-based similarity, add a data smoothing process to improve the prediction accuracy, and extract information from the QoS of similar services to similar users to handle the data *sparsity* problem. In addition, a TNS strategy has been proposed to improve the scalability of ADF. The experiments based on a public data set prove that *ADF* outperforms the existing methods.

In future work, we will explore other directions of our proposed CF-based service selection architecture, e.g., pruning. Moreover, We will collect more QoS data from Web services to perform larger scale experiments. Finally, tags, which are annotated to Web services by users, will be utilized to improve the performance of CF-based service selection.

REFERENCES

[1] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in *Proc. Int. World Wide Web Conf.*, Madrid, Spain, 2009, pp. 881–890.

[2] S. Ran, "A model for web services discovery with QoS," *ACM SIGecom Exchanges*, vol. 4, no. 1, pp. 1–10, Spring 2003.

[3] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[4] P. Xiong, Y. Fan, and M. Zhou, "QoS-aware web service configuration," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 888–895, Jul. 2008.

[5] P. Xiong and M. Zhou, "Web service configuration under multiple quality-of-service attributes," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 2, pp. 311–321, Apr. 2009.

[6] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world web services," in *Proc. Int. Conf. Web Serv.*, Miami, FL, 2010, pp. 83–90.

[7] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, Madison, WI, 1998, pp. 43–52.

[8] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction for web services via collaborative filtering," in *Proc. Int. Conf. Web Serv.*, Salt Lake City, UT, 2007, pp. 439–446.

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

[10] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized QoS-aware web service recommendation and visualization," *IEEE Trans. Serv. Comput.*, to be published.

[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. ACM SIGIR Conf.*, Berkeley, CA, 1999, pp. 230–237.

[12] R. Jin, J. Y. Chai, and L. Si, "An automatic weighting scheme for collaborative filtering," in *Proc. ACM SIGIR Conf.*, Sheffield, U.K., 2004, pp. 337–344.

[13] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, Jan. 2004.

[14] B. Sarwar, G. Karypic, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. Int. World Wide Web Conf.*, Hong Kong, China, 2001, pp. 285–295.

[15] J. Wang, A. P. Vries, and M. J. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proc. ACM SIGIR Conf.*, Seattle, WA, 2006, pp. 501–508.

[16] L. Si and R. Jin, "Flexible mixture model for collaborative filtering," in *Proc. Int. Conf. Mach. Learn.*, Washington, DC, 2003, pp. 704–711.

[17] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 89–115, Jan. 2004.

[18] T. Hofmann, "Collaborative filtering via guassian probabilistic latent semantic analysis," in *Proc. ACM SIGIR Conf.*, Toronto, ON, Canada, 2003, pp. 259–266.

[19] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *Amer. Stat.*, vol. 42, no. 1, pp. 59–66, Feb. 1988.

[20] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Symp. Math, Stat., Probability*, Berkeley, CA, 1967, pp. 281–297.

[21] L. Chen, J. Wu, R. Jia, S. Deng, and Y. Li, "Recommendation on uncertain services," in *Proc. Int. Conf. Web Serv.*, Miami, FL, 2010, pp. 683–684.

[22] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 369–384, Jun. 2007.

[23] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. Int. Conf. Data Eng.*, Heidelberg, Germany, 2001, pp. 421–430.

[24] E. Al-Masri and Q. H. Mahmoud, "Discovering the best web service," in *Proc. International World Wide Web Conf.*, Banff, AB, Canada, 2007, pp. 1257–1258.

[25] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end QoS constraints," *ACM Trans. Web*, vol. 1, no. 1, pp. 1–26, May 2007.

[26] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti, "Flow-based service selection for web service composition supporting multiple QoS classes," in *Proc. Int. Conf. Web Serv.*, Salt Lake City, UT, 2007, pp. 743–750.

[27] M. Papazoglou, "Service-oriented computing: concepts, characteristics and directions," in *Proc. Int. Conf. Web Inf. Syst. Eng.*, Roma, Italy, 2003, pp. 3–12.

[28] Y. Liu, A. H. Ngu, and L. Zeng, "QoS computation and policing in dynamic web service selection," in *Proc. Int. World Wide Web Conf.*, Manhattan, NY, 2004, pp. 66–73.

[29] Y. Taher, D. Benslimane, M.-C. Fauvet, and Z. Maamar, "Towards an approach for web services substitution," in *Proc. IDEAS*, 2006, pp. 166–173.

[30] D. Kossmann, F. Ravade, and D. Abugov, "Quadtree and r-tree indexes in oracle spatial: A comparison using gis data," in *Proc. SIGMOD*, Madison, WI, 2002, pp. 546–557.

[31] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimala and progressive algorithm for skyline queries," in *Proc. SIGMOD*, San Diego, CA, 2003, pp. 467–478.

[32] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng, "Quality driven web services composition," in *Proc. Int. World Wide Web Conf.*, Budapest, Hungary, 2003, pp. 411–421.

[33] M. Jahrer, A. Toshcer, and R. Legenstein, "Combining predictions for accurate recommender systems," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, Washington, DC, 2010, pp. 285–295.

[34] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proc. ACM SIGIR Conf.*, Salvador, Brazil, 2005, pp. 114–121.

[35] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 355–369, Mar. 2007.

[36] R. Nayak, "Data mining in web services discovery and monitoring," *Int. J. Web Serv. Res.*, vol. 5, no. 1, pp. 62–80, 2008.

[37] P. Plebani and B. Pernici, "Urbe: Web service retrieval based on similarity evaluation," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 11, pp. 1629–1642, Nov. 2009.

[38] L.-H. Vu, M. Hauswirth, and K. Aberer, "QoS-based service selection and ranking with trust and reputation management," in *Proc. Int. Conf. Cooperative Inf. Syst.*, 2005, pp. 466–483.

[39] L.-J. Zhang, S. Cheng, C. K. Chang, and Q. Zhou, "A pattern-recognition-based algorithm and case study for clustering and selecting business services," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 42, no. 1, pp. 102–114, Jan. 2012.

[40] P. Xiong, Y. Fan, and M. Zhou, "A petri net approach to analysis and composition of web services," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 2, pp. 376–387, Mar. 2010.

[41] Z. Zheng, Y. Zhang, and M. R. Lyu, "CloudRank: A QoS-driven component ranking framework for cloud computing," in *Proc. IEEE SRDS*, 2010, pp. 184–193.

[42] L. Chen, Y. Feng, J. Wu, and Z. Zheng, "An enhanced QoS prediction approach for service selection," in *Proc. Int. Conf. Serv. Comput.*, 2011, pp. 727–728.

Jian Wu (M'05) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 1998 and 2004, respectively.

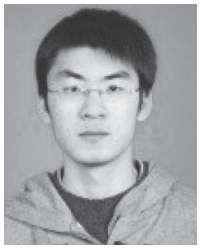
He is currently an Associate Professor with the College of Computer Science, Zhejiang University, and a Visiting Professor with the University of Illinois at Urbana-Champaign, Urbana. He is currently leading some research projects supported by National Natural Science Foundation of China and National High-tech R&D Program of China (863 Program). His research interests include service computing and data mining.

Dr. Wu was the recipient of the Second Grade Prize of the National Science Progress Award.

Liang Chen (S'12) received the B.S. degree in computer science from Zhejiang University, Hangzhou, China, in 2009. He is currently working toward the Ph.D. degree in the College of Computer Science, Zhejiang University.

His publications have appeared in some well-known conference proceedings and international journals. He also served as a Reviewer for some international conferences and journals. His research interests include service computing and data mining.

Mr. Chen received the award of Excellent Intern from Microsoft Research Asia in 2010.



Yipeng Feng received the B.S. degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He is currently working toward the Master's degree in the College of Computer Science, Zheijiang University.

His research interests include recommender systems, data mining, and service computing.

Zibin Zheng (S'07–M'11) received the Ph.D. degree from The Chinese University of Hong Kong (CUHK), Shatin, Hong Kong, in 2010.

He is currently an Associate Research Fellow with the Shenzhen Research Institute, CUHK. His research interests include cloud computing, service computing, and software engineering.

Dr. Zheng served as a program committee member of IEEE CLOUD2009, CLOUDCOMPUTING2010-2012, CGC2011, and SCC2011-2012. He was the recipient of the ACM SIGSOFT Distinguished Paper Award at ICSE2010, the Best Student Paper Award at ICWS2010, the First Runner-up Award at the 2010 IEEE Hong Kong Postgraduate Student Research Paper Competition, the 2010–2011 IBM Ph.D. Fellowship Award, and the Outstanding Thesis Award of CUHK in 2012.

Meng Chu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in electrical engineering from Nanjing University of Science and Technology, Nanjing, China, in 1983; the M.S. degree in automatic control from Beijing Institute of Technology, Beijing, China, in 1986; and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1990.

He is currently a Professor of electrical and computer engineering and a Director with the Discrete-Event Systems Laboratory, New Jersey Institute of Technology (NJIT), Newark. He has over 440 publications including ten books, 200+ journal papers (majority in IEEE Transactions), and 18 book chapters. His research interests include Petri nets, computer-integrated systems, wireless ad hoc and sensor networks, semiconductor manufacturing, and energy systems.

Dr. Zhou is currently the Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING and the Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: PART A and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He is a Founding Editor of the IEEE Book Series of System Science and Engineering. He is a Life Member of the Chinese Association for Science and Technology–USA and served as its President in 1999.

Zhaohui Wu (SM'05) received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 1993.

From 1991 to 1993, he was a joint Ph.D. student in the area of knowledge representation and expert system with the German Research Center for Artificial Intelligence, Kaiserslautern, Germany. He is currently a Professor with the College of Computer Science and a Director with the Institute of Computer System and Architecture, Zhejiang University. He is the author of four books and more than 100 referred papers. His research interests include intelligent transportation systems, distributed artificial intelligence, semantic grids, and ubiquitous embedded systems.

Dr. Wu is a Standing Council Member of China Computer Federation (CCF), Beijing. Since June 2005, he has been the Vice Chair of the CCF Pervasive Computing Committee. He is also on the editorial boards of several journals and has served as a Program Chair for various international conferences.