Applied Mathematics & Information Sciences
An International Journal

Towards QoS Prediction for Web Services based on Adjusted Euclidean Distances

Yuyu Yin[1], *Dongjing Yu*[1] *and Ying Li*[2]

[1] College of Computer, Hangzhou Dianzi University, P.R.China
[2] College of Computer Science and Technology, Zhejiang University, P.R.China

Received: May 02, 2012; Revised Sep. 25, 2012; Accepted Oct. 12, 2012
Published online: 1 Mar. 2013

Abstract: QoS-based service selection is an important issue of service-oriented computing (SOC). A common premise of previous research is that the QoS values of services to target users are supposed all known. However, the real situation is that many of them may be missing. In this paper, we propose an enhanced CF-based QoS prediction approach to predict such missing values. Compared with existing QoS prediction methods, our proposed one has three new features: 1) adding a data normalization process to remove the impact of different QoS scale; 2) using the adjusted Euclidean Distances equation for similarity calculation to improve the prediction accuracy; and 3) using a fusion approach to predict the missing values from two sources(user and service based). An extensive performance study based on a real public dataset is reported to verify its effectiveness.

Keywords: Quality of Services, Prediction, Service Selection.

1. Introduction

Service-oriented computing (SOC) paradigm and its realization through standardized web service technologies provide a promising solution for the seam-less integration of single-function applications to create new large-grained and value-added services. SOC attracts industrial's attention and is applied in many domains, e.g., workflow management, finances, and e-business. 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. Recently, there are a number of studies [1–3] 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 real world. Due to some inevitable elements, 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 /parseFeaureAASequenceFromFSOLVText, located in Spain) is $5626ms$, while that for user u_b(IP:`133.1.74.162`, Japan) to invoke the same one is $687ms$. A user can hardly have invoked all services, meaning that the QoS values of some services that the user has not invoked are missing. Hence, effective approaches are urgently needed to provide accurate prediction of the QoS values of different Web services for each user without requiring real Web service invocations.

In recent years, researchers have proposed a number of QoS prediction approaches [6–9]. Inspired by the application of Collaborative Filtering(CF) [7] in product recommendation, CF has been employed to predict QoS values using Web service QoS evaluations from different users. Moreover, extensive experimental studies have shown that CF-based approaches can obtain good overall prediction precision. However, the CF-based approaches can be restricted under dense past QoS values. As can be seen from [9], the more sparse QoS values is, the more greatly prediction precision degrades. In fact, QoS values may be very sparse since users may only invoke a few of candidate services each time.

To improve the prediction accuracy under sparse QoS values, we propose an enhanced CF-based approach to predict the missing QoS values. The basic idea is that Web

* Corresponding author: e-mail: yinyuyu@hdu.edu.cn

service QoS values of a user can be predicted by employing past Web service QoS values of other users. As different QoS scale can impact the prediction accuracy, our approaches first normalize to make QoS values in different scale be fair on making prediction. Then, considering overestimation of the user similarities using Pearson Correlation Coefficient(PCC), we propose an improved Euclidean Distances by using the standard derivation of users or services. Finally, the final QoS is estimated by fusing predictions from two sources: predictions based on QoS of the same service to similar users and those based on QoS of similar services to the same user.

The contributions of this work are as follows: (1) We propose an enhanced CF-based QoS prediction approach via adjusted Euclidean Distances. Our approach significantly enhances the prediction accuracy under sparse QoS values; (2) We evaluate the proposed approach experimentally by employing a real-world Web service QoS dataset. It contains the records of $1,974,675$ Web service invocations executed by 339 distributed service users on 5825 Web services.

The rest of this paper is organized as follows: Section 2 presents our QoS value prediction approach. Section 3 describes our experiments. Section 4 introduces related work and Section 5 concludes the paper.

2. Our Prediction Approach

2.1. Approach Outline

Our approach consists of four key procedures: 1) Normalization, 2) Similarity Computation, 3). Similarity Neighbor selection, and 4) QoS Prediction. Normalization makes QoS values into a uniform scope based on Gaussian approach [2]. Similarity Computation calculates the user and service similarity with adjusted Euclidean Distances based on their historical experiences. Similarity Neighbor selection uses the traditional Top-K algorithm to select the similar services or users. QoS Prediction is used to predict the missing values of services according to the existing QoS values by using our approach. For the convenience of our readers, Table 1 summarizes all important notations used in this paper.

2.2. Normalization

After analyzing real-world Web service QoS values, we find QoS value can be data type or ratio type. The value of ratio type property varies in a limited range, such as $0-100\%$, while the value of data type property may have quite different scale. A typical data type is response time, which is possible to vary in range of $[0s, 1s]$ for one kind of consumers, but in range of $[10s, 20s]$ for the other king of consumers.

The differences in QoS scale can impact user and service similarity. To solve the problem, Data type should be normalized to make data in different range be fair on making prediction. Therefore, Gaussian approach [2] is introduced to normalize QoS values from each user separately as shown in Equation(1), where $\bar{r}_{u_i}$ denotes the arithmetic mean of QoS value from user u_i, σ_i is the standard deviation of QoS values from u_i. We use $3\sigma_i$ because of the $3-\sigma$ rule, which helps to normalize the value into the range of $[0, 1]$.

$$r_{u_i,s_j} = 0.5 + \frac{r_{u_i,s_j} - \bar{r}_{u_i}}{2 \times 3\sigma_i} \quad (1)$$

Given a dataset consisting of M service users and N Web services, the invocation records between users and services can be denoted by a $M \times N$ matrix, called the user-service matrix. A user-service matrix example is given in Table 2. Every entry in this matrix r_{u_1,s_1} represents a record of invocation (QoS values, e.g., response time and throughput). For example, the response time for user u_1 to invoke service s_1 is $100ms$, and the throughput is 0.13. Then the vector r_{u_1,s_1} is $(100, 0.13)$ if we consider these two QoS parameters only. If user u_1 has not invoked the service s_3, then $r_{u_1,s_3} = null$.

Table 1 Key notations and their descriptions

Notation	Definition and Brief Description
u_1	a user
s_1	a Web service
S_{u_1}	the set of Web services have been invoked by u_1
US_1	the set of users have invoked s_1
r_{u_1,s_1}	the QoS of s_1 to u_1
ϕ	similarity between objects
P	prediction result
L	the set of similar objects
con	coefficient weight
T	the number of users in training data
g	the number of Web services invoked by the user in testing data

Table 2 user-service matrix

	s_1	s_2	s_3
u_1	$\langle 0.4s, 0.13\rangle$	$\langle 1.6s, 0.23\rangle$	?
u_2	$\langle 2.6s, 1.44\rangle$	$Null$	$\langle 3.5s, 6.12\rangle$
u_3	$\langle 0.8s, 0.45\rangle$	$\langle 0.9s, 6.55\rangle$	$\langle 5.1s, 7.12\rangle$
u_4	$\langle 8s, 0.8\rangle$	$\langle 3.0s, 0.92\rangle$	$Null$
u_5	$\langle 4.3s, 3.3\rangle$	$\langle 3.5s, 2.3\rangle$	$\langle 1.6s, 0.95\rangle$

2.3. Similarity Computation

This section introduces the similarity computation method of different service users as well as different Web services. Existing work [7–9] about QoS prediction uses PCC to compute the similarity of users or services. In this paper, Euclidean Distances is introduced to define the similarity of users or services.

After analyzing real-world Web service QoS values, we find the similarity of users or services is impacted by them itself. For example, due to the reason of a fast network, the response time of service s_i to all users, which invoked it, is lower than $200ms$. In such situations, the similarity between these users is larger using Euclidean Distances. As a result, it will overestimate the similarities of users who are actually not similar but happen to have similar QoS experience on a few co-invoked Web services [4]. Also, this will impact the final prediction accuracy. In order to address the problem, we adjust the traditional Euclidean Distances by using d.

The similarity between two users u_1 and u_2 based on the services they commonly invoke is computed using the following equation:

$$\phi_{u_1,u_2} = \frac{1}{\sqrt{\frac{\sum_{s\in S}(r_{u_1,s}-r_{u_2,s})^2}{|S|}}} \times d_s \tag{2}$$

Where $S^k = {S_{u_1}}^k \cap {S_{u_2}}^k$ is the set of services that are both invoked by user u_1 and u_2, $r_{u_1,s}$ is the vector of QoS values of service s invoked by user u_1, $r_{u_2,s}$ is the vector of QoS values of service s invoked by user u_2, and d_s is the standard deviation of QoS values of service s.

In addition, the similarity between services is computed using the following equation:

$$\phi_{s_1,s_2} = \frac{1}{\sqrt{\frac{\sum_{u\in U}(r_{u,s_1}-r_{u,s_2})^2}{|U|}}} \times d_u \tag{3}$$

where $U = U_{s_1} \cap U_{s_2}$ 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, r_{u,s_2} is the vector of QoS values of service s_2 invoked by u, and d_s is the standard deviation of QoS values from user u.

2.4. Similar Neighbor Selection

In Section 2.3, we have calculated the similarities between different users or services, and then we can choose a set of similar neighbors for target users or services. The process of selecting similar neighbor is crucial for the accuracy of prediction because the prediction of a missing value depends on the corresponding values of similar neighbor. In existing works [9], using PCC, the similarity between users or services is in the range of $[-1, 1]$ with a larger value indicating that u_1 and u_2 are more similar. PCC similarities smaller or equal to 0 will be excluded. Hence, traditional Top-K algorithm is not suitable for this scenario. Different from previous research, similarities is only larger or equal to 0 using our adjusted Euclidean Distances. Therefore, we use the traditional Top-K algorithm to select the similar neighbors.

2.5. QoS Prediction

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

$$P_u = \sum_{u_1\in L_u} \frac{\phi_{u,u_1} \times r_{u_1,s}}{\sum_{u_1\in L_u}\phi_{u,u_1}} \tag{4}$$

Where P_u is a vector of predicted QoS values of the missing value $r_{u_1,s}$ in the user-item matrix, L_u is u's similar users. The equation of service-based prediction is as follows.

$$P_s = \sum_{s_1\in L_s} \frac{\phi_{s,s_1} \times r_{u,s_1}}{\sum_{s_1\in L_s}\phi_{s,s_1}} \tag{5}$$

When L_u and L_s is not empty, predicting the missing value only with user-based methods or services-based methods will potentially ignore that the information of the similar users and similar services. They can make the prediction more accurate. Therefore, the final QoS is estimated by combining user-based method and service-based method.

Since these two predicted results are all computed based on similarities, we should compute the confidence weight of each predicted result to balance the results from these two prediction methods. The confidence weight of user-based prediction is defined as:

$$con_u = \frac{\sum_{u_1\in L_u}\phi_{u,u_1} \times \phi_{u,u_1}}{\sum_{u_1\in L_u}\phi_{u,u_1}} \tag{6}$$

The confidence weight of service-based prediction is:

$$con_s = \frac{\sum_{s_1\in L_s}\phi_{s,s_1} \times \phi_{s,s_1}}{\sum_{s_1\in L_s}\phi_{s,s_1}} \tag{7}$$

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 predicted result is the aggregation of user and service based predicted results, we set the parameters $w \in [0, 1]$ to determine how our QoS prediction relies on each result. The final equation for QoS prediction is as follows:

$$P(r_{u,s}) = w_u * P_u + w_s * P_s \tag{8}$$

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

$$w_u = \frac{con_u * w}{con_s * (1-w) + con_u * w} \tag{9}$$

and w_s is defined as:

$$w_s = \frac{con_s * (1-w)}{con_s * (1-w) + con_u * w} \tag{10}$$

The parameter w, means the participation that the corresponding prediction result takes in the final result. If $0 < w < 1$, our prediction approach aggregate the final results from user and service based predicted results. If $w = 1$, our approach turns to be the user-based prediction approach. Similarly, if $w = 0$, our approach degrades to the service-based approach.

3. Experiments

In this section, we present an experimental evaluation of the proposed approach by measuring (a) performance of different approaches (UPCC, IPCC, WSRec, and Our Approach), in terms of Normalized Mean Absolute Error (NMAE); (b) Impact of the adjusted Euclidean Distances;(c) impact of the confidence weight; and (d) impact of parameter w.

3.1. Experimental Setup

We have conducted our experiments using a public real-worldWeb service QoS dataset, which is collected by Zibin Zheng et.al [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 2 parameters: Response Time and Throughput. More details about this dataset can be found in [6]. In this paper, we randomly extract 150 users, 100 Web services and use the invocation records between them as the experimental data.

Our experiments are implemented with Matlab 7.0 and mysql 5.0. They are conducted on a Dell Inspire R13 machine with a 2.27 GHz Intel Core I5 CPU and 2GB RAM, running Windows 7 OS.

3.2. Evaluation Metric

In our experiments, $NMAE$ is used to evaluate the accuracy of prediction. Mean Absolute Error (MAE) is as follows:

$$MAE = \frac{\sum_{U,S} |r_{u,s} - \hat{r}_{u,s}|}{N} \quad (11)$$

where $r_{u,s}$ represents 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, services QoS value range may differ so tremendously that only MAE is not objective enough. As an adjustment, $NMAE$ normalizes the differences range of MAE by computing:

$$NMAE = \frac{MAE}{\sum_{U,S} \frac{r_{u,s}}{N}} \quad (12)$$

The smaller $NMAE$, the more accurate QoS prediction.

3.3. Performance Comparison

We compare the proposed approach with three famous prediction methods: User-based algorithm using PCC(UPCC), Item-based algorithm using PCC(IPCC), and WSRec [9]. Note that Equation (4) and (5) are employed to calculate UPCC and IPCC, respectively. To the best of our knowledge, WSRec approach is the best one for QoS prediction at present.

As the dataset used for experiments is the set of invocation records between 150 users and 100 Web services, we create a 150×100 user-service matrix, where each entry in it is a vector including two QoS values: Response Time, Throughput. During the experiment, the 150×100 matrix is divided into two parts, N rows as the training matrix and the other $(150 - N)$ rows as the testing one. The users in 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 situation of experiments similar to the real scenario. In addition, we vary the number of invocation records that target users can provide(in other word, the number of Web services that target users have invoked). To minimize error, each experiment is looped 50 times and the average value is reported.

Table 3 shows the prediction performance of above four approaches on Response Time and Throughput employing 5%, 10%, 15%, and 20% density of the training matrix respectively. For the users in testing matrix (target users), we vary the number of invoked Web services as 10, 20, and 30 by randomly sampling (named as $g10$, $g20$, $g30$ in Table 3). 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 $w = 0.5$, and $K = 10$ (the number of similar neighbors in Top-K Algorithm). From Table 3, we find that our prediction approach obtains smaller $NMAE$ values, which means higher prediction accuracy in all cases, especially under the training matrix is sparse. This demonstrates that our approach gives more accurate prediction. Comparing the prediction results for cases of $T = 100$ and 140, we can find that the latters $NMAE$ values are smaller, which indicates that the increase of T improves the prediction accuracy. Similarly, the increase of the density of a training matrix improves the accuracy. It can be easily explained as higher density means more training data. Furthermore, the increase of the number of invoked services ($g10$, $g20$, and $g30$) also improves the prediction accuracy.

3.4. Evaluation of the adjusted Euclidean Distances

To study its impact to the performance of the adjusted Euclidean Distances, we implement two versions of our prediction approach, one version employs the traditional Euclidean Distances and the other employs the adjusted Euclidean Distances for the similarity calculation. In the experiment, we set $w = 0.5$ and $K = 10$. Figure 1(a)-(d) shows the performance of the two versions on the training matrix of Response Time when T is 100 and 140, while Figure 1(e)-(h) shows that on the training matrix of

Table 3 Comparison of Prediction Accuracy (a smaller value means a better performance)

Density	**Methods**	**T=100**					
		Response Time			**Throughput**		
		***g*10**	***g*20**	***g*30**	***g*10**	***g*20**	***g*30**
5%	**UPCC**	0.609	0.5625	0.5534	0.8949	0.8369	0.8201
	IPCC	0.5749	0.5552	0.548	0.8286	0.8261	0.8204
	WSRec	0.5212	0.5107	0.4902	0.7832	0.7721	0.7716
	OurApproach	0.4391	0.4089	0.3794	0.67	0.6611	0.6604
10%	**UPCC**	0.5543	0.5479	0.547	0.8391	0.8233	0.8167
	IPCC	0.4746	0.4512	0.4234	0.7629	0.7363	0.7124
	WSRec	0.4404	0.404	0.3855	0.7329	0.6955	0.6818
	OurApproach	0.3552	0.3144	0.3009	0.6295	0.5949	0.5793
15%	**UPCC**	0.5195	0.5061	0.4903	0.7878	0.7746	0.7717
	IPCC	0.4385	0.4121	0.411	0.7242	0.7182	0.6712
	WSRec	0.3925	0.3855	0.3738	0.698	0.6776	0.6393
	OurApproach	0.321	0.2907	0.2854	0.5943	0.5864	0.5719
20%	**UPCC**	0.4903	0.4852	0.4733	0.7554	0.7504	0.7414
	IPCC	0.4093	0.3767	0.3674	0.701	0.6971	0.6625
	WSRec	0.3653	0.3411	0.3305	0.6479	0.6377	0.6211
	OurApproach	0.3107	0.2869	0.2792	0.5589	0.5356	0.5213
Density	**Methods**	**T=140**					
		Response Time			**Throughput**		
		***g*10**	***g*20**	***g*30**	***g*10**	***g*20**	***g*30**
5%	**UPCC**	0.6103	0.5717	0.5497	0.9199	0.8171	0.8082
	IPCC	0.5531	0.5348	0.5241	0.8548	0.8243	0.7688
	WSRec	0.5036	0.4808	0.4614	0.7813	0.7423	0.7158
	OurApproach	0.389	0.3921	0.3858	0.6873	0.636	0.6085
10%	**UPCC**	0.5486	0.5399	0.5019	0.8255	0.7953	0.7791
	IPCC	0.4626	0.4358	0.4225	0.7388	0.6769	0.6525
	WSRec	0.4138	0.3914	0.3884	0.706	0.618	0.6176
	OurApproach	0.3277	0.3127	0.2899	0.6168	0.5092	0.5001
15%	**UPCC**	0.4864	0.4711	0.4505	0.7946	0.7659	0.7261
	IPCC	0.3976	0.377	0.3635	0.6971	0.6432	0.6397
	WSRec	0.3539	0.3289	0.323	0.6757	0.6528	0.6078
	OurApproach	0.2986	0.2591	0.2543	0.5727	0.5658	0.5365
20%	**UPCC**	0.4792	0.4602	0.4433	0.7492	0.7405	0.7171
	IPCC	0.3703	0.3498	0.3406	0.6709	0.6362	0.6297
	WSRec	0.3278	0.3136	0.3028	0.6382	0.5849	0.5731
	OurApproach	0.2775	0.2639	0.2476	0.5209	0.483	0.4799

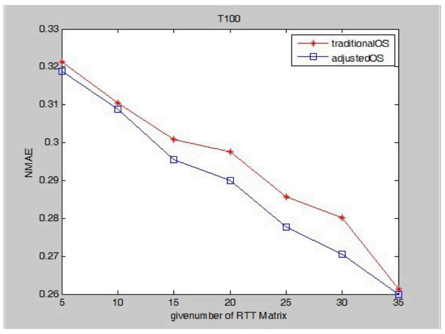
Throughput. Note that the word Given number in below figures means g, and the scale of y-coordinate(it denotes $NMAE$) has small difference in Figure 1, Figure 2, and Figure 3 for accurately showing the value of $NMAE$.

In Figure 1(a)-(b), g is fixed as 10. From Figure 1(a), with the density of a training matrix varying from 5% to 40%, we can find that the prediction approach with the adjusted Euclidean Distances outperforms the prediction approach with the traditional Euclidean Distances. Figure 1(b) shows the performance comparison when the number of T is 140, and the result is similar to Figure1(a). In Figure 1(c)-(d), the density of the training matrix is fixed as 10%. From Figure 1(c), we can find that the result is similar to Figure 1(a)-(b) with g varying from 5 to 35. When the number of T is 140, the improvement that the adjusted Euclidean Distances brings to our prediction approach is clearer. Moreover, from Figure 1(e)-(f), we can draw the similar conclusion with Figure 1(a)-(d).

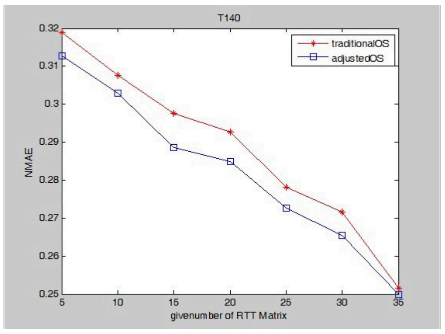
Figure 1 shows that our prediction approach obtains better prediction accuracy. Clearly, the adjusted Euclidean Distancess usage enhances the accuracy of prediction.

3.5. Impact of the confidence weight

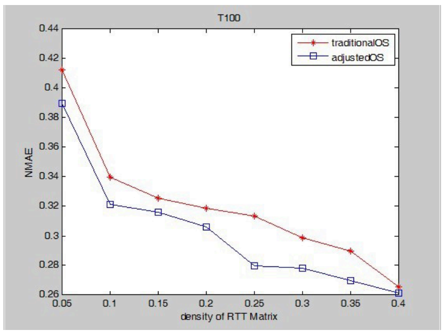
Confidence weight determines how to make use of the predicted results from the user-based method and the service-based method. To study its impact, we also implement two versions of our prediction approach, one version employs confidence weight, while the other version does not. In the experiments, $w = 0.5$, $T = 140$, and $K = 10$. Figure 2(b) and (d) show the trend of Response Time and Throughput with given number change, while Figure 2(a) and (c) show



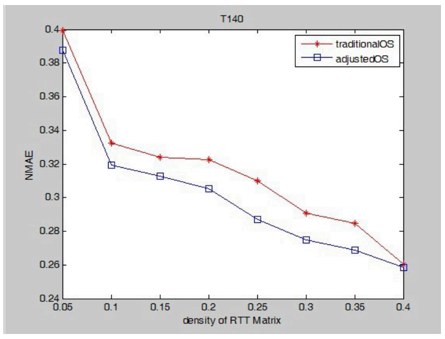
(a)



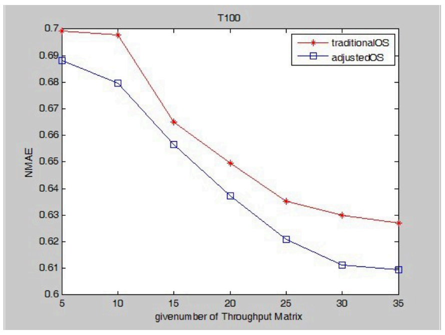
(b)



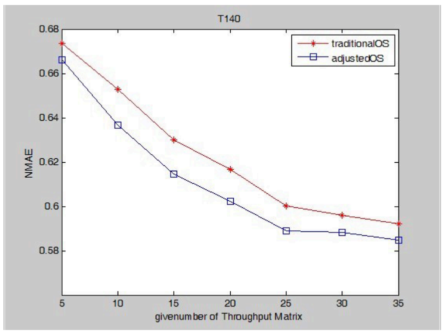
(c)



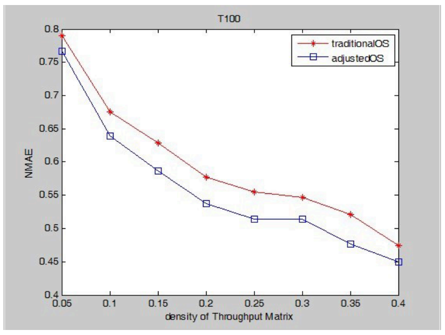
(d)



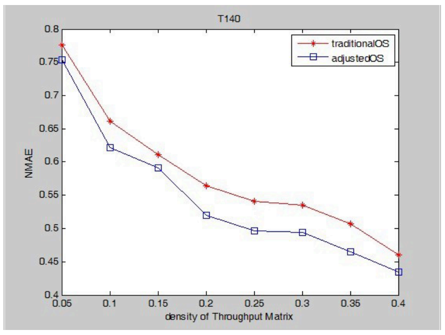
(e)



(f)

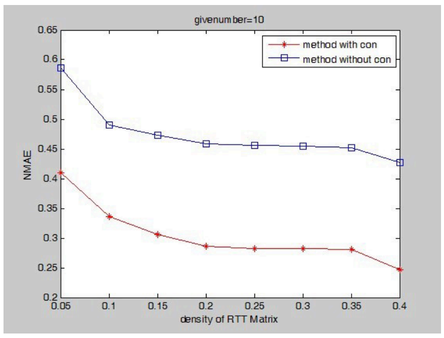


(g)

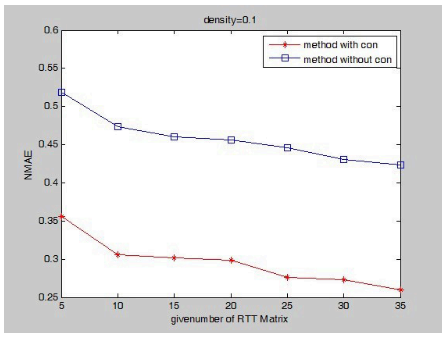


(h)

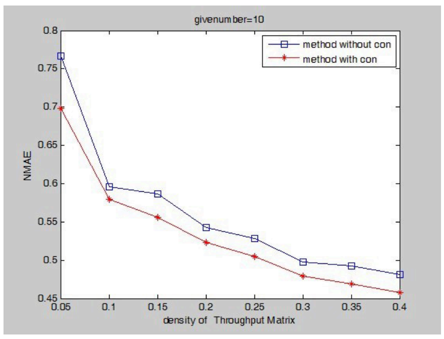
Figure 1 Impact of the Adjusted Euclidean Distances



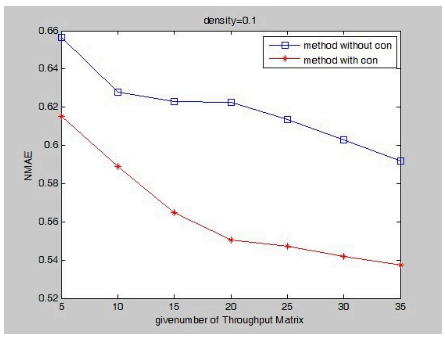
(a)



(b)



(c)



(d)

Figure 2 Impact of the confidence weight

the trend of Response Time and Throughput with training matrix density change. As shown in Figure 2, the version with confidence weight outperforms the version without confidence weight for both Response Time and Throughput.

3.6. Impact of w

The value of w stands for the weight of user or service based prediction. w makes our prediction method more feasible and adaptable to different data sets. To study its impact, we set $K = 10$ and $T = 140$. We vary the value of w from 0 to 1 with a step value of 0.1. Figure 3(a) and 3(c) show the results of $g = 10$; 20, and 30 with 10% density training matrix of response time and throughput, respectively. Figs. 3(b) and 3(d) show the results of $g = 10$; 20, and 30 of response time and throughput, with 20% density training matrix of response time and throughput, respectively.

From Figure 3, we draw the conclusion that an optimal w value improves the accuracy of the prediction while an unsuitable value makes it worse. Another interesting observation is that, in Figure 3(a), with g increasing from 10 to 30, the optimal w value shifts from 0.5 to 0.7. In Figure 3(b), the optimal w value shifts from 0.6 to 0.8 as g varies from 10 to 30.This indicates that the optimal w value is influenced by the given number. In Figure 3(d), the optimal value is 0.5. Therefore, the optimal value is not influenced by g but influenced by the nature of datasets for this example.

4. Related Work

Some important problems have been widely discussed, for example, reliability, substitutability, and adaption of service. The problem of QoS-based service selection also attracts many researchers attentions and was discussed in a number of studies recently. A common premise of previous research is that the QoS values of services to target users are all known. However, there are often missing QoS values of services to the consumer in a real situation. Therefore, a fundamental process before QoS-based service selection is to predict such missing values.

In this paper, we use a Collaborative Filtering based approach to handle this problem. It is often classified as memory-based or model-based one. In the memory-based one, all training data are stored in memory. In the prediction phase, similar objects (users or items) are sorted based on their similarities with the active object. Pearson Correlation Coefficient is 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 of memory-based methods

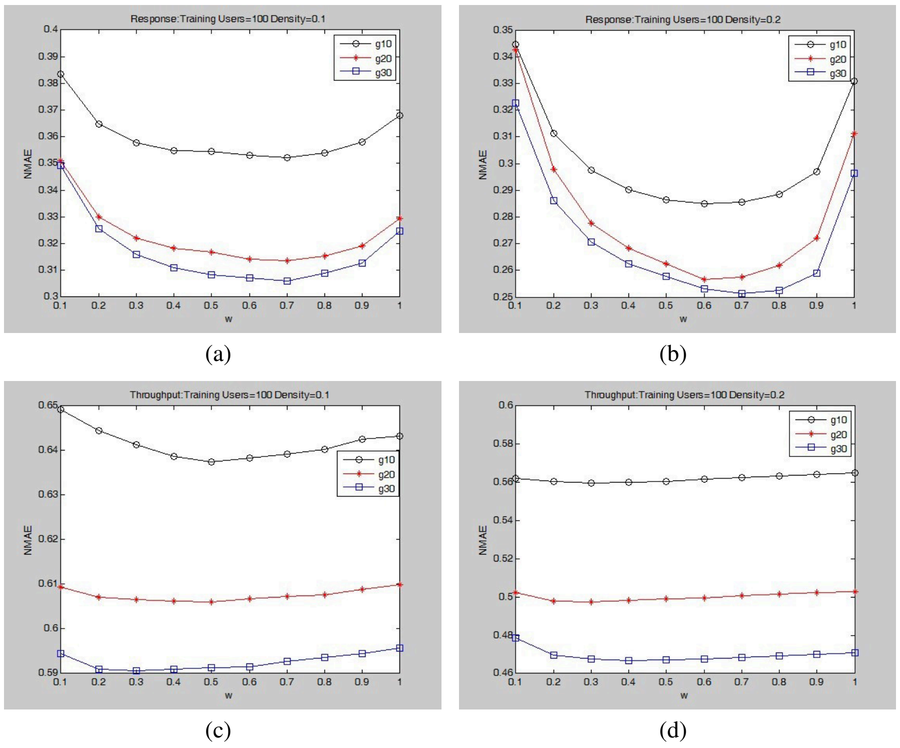


Figure 3 Impact of w

include user-based methods [12–14], item-based methods [15,16] and fusion methods [17]. The advantage of the memory-based methods is that they are simple and intuitive on a conceptual level while avoiding the complications of a potentially expensive model-building stage. Their drawbacks include: 1) scalability is not well; and 2) nothing is really learned from the available user profiles and very little general insight is gained. In the model-based approach, training data are used to generate a predicting model that is able to predict the missing data. The most analyzed examples include decision tree [12], aspect models [18], latent semantic models [11,19].

Limited work has been done to predict the missing QoS values. Shao et al. propose a user-based Collaborative Filtering algorithm to make similarity mining and predict the QoS of Web services from consumers experiences [8]. Zheng et al. present a hybrid approach which combines user-based and item-based approach together to predict the QoS of Web services [9]. They employ two confidence weights to balance these two predicted values. In Section 3, we have shown that our approach outperforms the WSRec approach. Chen et al. discover the great influence of a users location to the accuracy of prediction and propose a region-based hybrid Collaborative Filtering algorithm to predict the QoS of services [10].

In the paper, we also propose CF-based prediction to handle the QoS prediction problem. Before similarity computation, we introduce a data normalization process to remove the impact of different QoS scales. In the process of similarity computation, we use the adjusted Euclidean Distances equation to compute the service similarity to improve the prediction accuracy. In the process of result generation, we use a fusion approach to generate the final result from two sources (user and service based).

5. Conclusion and Future Work

In this paper, we also present a CF-based prediction approach to predict the missing QoS values. Different from the previous methods, we add a data normalization process to remove the impact of different QoS scales, adjust the traditional Euclidean Distances by using standard derivation of service and user to compute the service-based similarity, and combine the user-based approach and the service-based approach to predict the final results. The experiments based on a public dataset prove that our prediction approach outperforms the existing methods.

In our future work, the other parts in our CF-based service selection framework will be explored. Further, we will collect more QoS data of Web services to improve the scale of our experiments.

Acknowledgement

This paper is granted by National Natural Science Foundation of China under Grant(No. 61100043), Zhejiang Provincial Natural Science Foundation(No.LY12F02003), and The National Key Technology R&D Program under Grant(No. 2012BAH24B04).

References

[1] M. Alrifai and T. Risse, Combining global optimization with local selection for efficient qos-aware service composition. Proc. of International World Wide Web Conference, Madrid, Spain, 2009: 881-890.

[2] S. Ran, A model for web services discovery with qos, ACM SIGecom Exchanges, 2003: 1-10.

[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. on Software Engineering, 2004, 5: 311-327.

[4] Xiong, P. C., Y. S. Fan, and M. C. Zhou, Qos-aware web service configuration, IEEE Trans. on Systems, Man, and Cybernetics: Part A, 2008 (38), 4: 888-895.

[5] Xiong, P. C., and M. C. Zhou, Web service configuration under multiple quality-of-service attributes, IEEE Trans. On Automation Science and Engineering, 2009 (6), 2: 311-321.

[6] Z. Zheng, Y. Zhang, and M. R. Lyu, Distributed qos evaluation for real-world web services, Proc. of International Conference on Web Services, Miami, USA, 2010; 83-90.

[7] J. S. Breese, D. Heckerman, and C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, Proc. of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, USA, 1998: 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, Proc. of International Conference on Web Services, Salt Lake City, USA, 2007: 439-446.

[9] Z. Zheng, H. Ma, M. R. Lyu, and I. King, Qos-aware web service recommendation by collaborative filtering, IEEE Transactions on Service Computing, 2011 (4), 2: 140-152.

[10] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, Personalized qos-aware web service recommendation and visualization, IEEE Transactions on Services Computing, 2011 (99). PrePrints.

[11] Hofmann, T., Collaborative filtering via guassian probabilistic latent semantic analysis, ACM SIGIR conference, Toronto,Canada, 2003: 259-266.

[12] J. S. Breese, D. Heckerman, and C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, Proc. Of Conference on Uncertainty in Artificial Intelligence, Madison, USA, 1998: 43-52.

[13] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, An algorithmic framework for performing collaborative filtering, ACM SIGIR Conference, Berkeley, USA, 1999: 230-237.

[14] R. Jin, J. Y. Chai, and L. Si, An automatic weighting scheme for collaborative filtering, ACM SIGIR Conference, Sheffield, UK, 2004: 337-344.

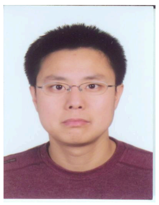
[15] M. Deshpande and G. Karypis, Item-based top-n recommendation algorithms, ACM Transaction on Information System, 2004(22),1: 143–177.

[16] B. Sarwar, G. Karypic, J. Konstan, and J. Riedl, Item-based collaborative filtering recommendation algorithms, Proc. Of International World Wide Web Conference, Hong Kong, China, 2001: 285-295.

[17] J. Wang, A. P. Vries, and M. J. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, ACM SIGIR Conference, Seattle, USA,2006: 501-508.

[18] L. Si and R. Jin, Flexible mixture model for collaborative filtering, Proc. of International Conference on Machine Learning, Washington, USA, 2003.

[19] T. Hofmann, Latent semantic models for collaborative filtering, ACM Transaction on Information System, 2004 (22), 1: 89-111.



Yuyu Yin received the Doctors degree in computer science from Zhejiang University, Hangzhou, China, in 2010. He is currently an assistant professor in Hangzhou Dianzi University. His research interests include service computing, cloud computing and middleware techniques.

Ying Li received the Doctors degree in software technology from Zhejiang University, Hangzhou, China, in 2000. He is currently an associate professor in Zhejiang University. His research interests include software architecture, software automation, compiling technology, and middleware techniques.