QoS Value Prediction Using a Combination of Filtering Method and Neural Network Regression

Soumi Chattopadhyay[1(✉)] and Ansuman Banerjee[2]

[1] Indian Institute of Information Technology, Guwahati, India
soumi61@gmail.com
[2] Indian Statistical Institute, Kolkata, India

Abstract. With increasing demand and adoption of web services in the world wide web, selecting an appropriate web service for recommendation is becoming a challenging problem to address today. The Quality of Service (QoS) parameters, which essentially represent the performance of a web service, play a crucial role in web service selection. However, obtaining the exact value of a QoS parameter of service before its execution is impossible, due to the variation of the QoS parameter across time and users. Therefore, predicting the value of a QoS parameter has attracted significant research attention. In this paper, we consider the QoS prediction problem and propose a novel solution by leveraging the past information of service invocations. Our proposal, on one hand, is a combination of collaborative filtering and neural network-based regression model. Our filtering approach, on the other hand, is a coalition of the user-intensive and service-intensive models. In the first step of our approach, we generate a set of similar users on a set of similar services. We then employ a neural network-based regression module to predict the QoS value of a target service for a target user. The experiments are conducted on the WS-DREAM public benchmark dataset. Experimental results show the superiority of our method over state-of-the-art approaches.

1 Introduction

With the proliferation of emerging technologies in the era of Internet-of-Things (IoT), the number of web services is increasing day by day. The existence of a large number of competing, functionally equivalent web services in world wide web, makes the problem of recommending an appropriate service for a specific task, quite challenging in recent times. A number of different factors may actually influence the process of recommendation [4,19,20]. The QoS parameter (e.g., response time, throughput, reliability, availability) being the representative of the performance of a web service is one of the key factors that may have an impact on service recommendation. However, the value of a QoS parameter of a web service varies across time and users. Therefore, obtaining the exact QoS that a user will witness during invocation is a difficult task. Prediction plays an important role in this context to obtain a close enough approximate QoS value

S. Yangui et al. (Eds.): ICSOC 2019, LNCS 11895, pp. 135–150, 2019.
https://doi.org/10.1007/978-3-030-33702-5_11

for recommendation. Quite evidently, the task of prediction is recognized as one of the fundamental research challenges in the domain of services computing.
In this paper, we address the problem of predicting the QoS value of a service for a given user by leveraging the past user-service QoS invocation profiles consisting of the QoS values of a set of services across different users. A significant number of research articles exist in literature which deal with this problem. Collaborative filtering [3,15] is one of the most popular methods adopted in this domain to predict the missing value. The collaborative filtering technique is classified into two categories: memory-based and model-based. The memory-based collaborative filtering comprises the computation of either the set of similar users [3] or the set of similar services [14] or the combination of them [25] followed by the computation of average QoS values and the computation of the deviation migration. However, these approaches suffer from the problem of the sparsity of the user-service invocation matrix. Therefore, model-based collaborative filtering is used which can deal with the sparsity problem. Matrix factorization [9,10,23] is a class of model-based collaborative filtering technique used for this problem. Though the contemporary approaches are able to predict the missing QoS value of a service for a target user, however, the prediction accuracy still is not quite satisfactory. Therefore, there is a scope for improving the prediction accuracy.

In this paper, we propose a new approach for predicting the QoS value of a service for a target user. Our method combines two primary techniques, i.e., collaborative filtering with a regression method, to come up with a solution. We first use the collaborative filtering technique to filter the set of users and services. Our filtering method is again a combination of the user-intensive and service-intensive filtering models. In user-intensive (service-intensive) filtering, we first find a set of similar users (services) from the given user-service invocation profile. We then find a set of similar services (users) from the user-service invocation profile corresponding to the set of users (services) obtained earlier. Once the filtering is done, we combine the results for further processing. Instead of computing the average QoS value and the deviation migration as done in the collaborative filtering approach, in our final step, we employ a neural network-based regression module to predict the QoS value of a service for a target user. We have shown the significance of each step of our proposal experimentally.

We have implemented our proposed framework and tested the performance of our approach on a public benchmark dataset, called WS-DREAM [24]. We have compared our method with state-of-the-art approaches. The experimental results show that our method achieves better performance in terms of accuracy as compared to others.

The contributions of this paper are summarized below:

(i) We propose a new approach for QoS prediction. On one side, our approach leverages the principle of collaborative filtering. On the other side, our approach takes advantage of the power of a neural network-based regression method.
(ii) We propose a filtering method, which is a combination of user-intensive and service-intensive models.
(iii) To find the set of similar users (and services), we propose a method based on unsupervised learning.

(iv) We have implemented our framework. A rigorous experiment has been conducted on the WS-DREAM dataset to establish our findings. The experimental results demonstrate that our method is more efficient in terms of prediction accuracy as compared to its contemporary approaches.

2 Related Work

A number of work [2,5,12,13,17] has been carried out in literature to address the problem of QoS value prediction. Collaborative filtering [15,16,21] technique is one of the key techniques used for the prediction. The collaborative filtering approach can be of two types: memory-based and model-based. The memory-based collaborative filtering approach uses the user-service invocation profile to find the set of similar users or services. Depending on the similarity finding method, the memory based collaborative filtering is again classified into two categories: user-intensive and service-intensive. In the user-intensive collaborative filtering method [3], a set of users similar to the target user is computed, while in the service-intensive filtering method [14], a set of services similar to the target service is computed. There are some research works [22,25] in literature, which combine both the user-intensive and service-intensive filtering techniques to obtain the predicted value. The main disadvantage of this approach is that the prediction accuracy decreases as data gets sparse. One possible solution to this problem is to employ model-based collaborative filtering. One such approach is matrix factorization [9,10,20,23], which is widely used to predict the QoS value of a service. In matrix factorization, the user-service QoS invocation matrix is decomposed into the product of two lower-dimensional rectangular matrices to improve the robustness and accuracy of the memory-based approach.

Although state-of-the-art approaches can predict the missing QoS values, however, they fail to achieve satisfactory prediction accuracy. Therefore, in this paper, we propose a novel approach to improve the prediction accuracy.

3 Overview and Problem Formulation

In this section, we formalize our problem statement. We begin with defining two terminologies as follows.

Definition 1 (QoS Invocation Log). *A QoS invocation log is defined as a 3-tuple* $(u_i, s_j, q_{i,j})$*, where* u_i *is a user,* s_j *is a web service and* $q_{i,j}$ *denotes the value of a given QoS parameter* q *when the user* u_i *invoked the service* s_j*.* ■

Once a user invokes a service, the corresponding invocation log is recorded. The QoS invocation logs are stored in the form of a matrix. We now define the concept of a QoS invocation log matrix.

Definition 2 (QoS Invocation Log Matrix). *The QoS invocation log matrix* $\mathcal{Q}$ *is a matrix with dimension* $n \times k$*, where* n *is the number of users and* k *is the number of web services. Each entry of the matrix* $\mathcal{Q}(i, j)$ *represents* $q_{i,j}$*.* ■

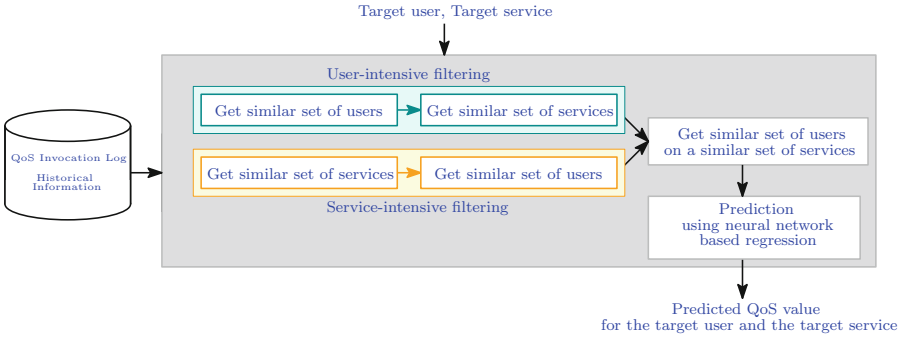

Fig. 1. Our proposed framework

Example 1. Consider $\mathcal{U} = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ be a set of 6 users and $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ be a set of 6 web services. Table 1 represents the QoS invocation log matrix $\mathcal{Q}$ for the set of users $\mathcal{U}$ and the set of services $\mathcal{S}$. $\mathcal{Q}(i, j)$ represents the value of the response time (in millisecond) of $s_j \in \mathcal{S}$ during the invocation of s_j by $u_i \in \mathcal{U}$. Our objective, here, is to predict the value of the QoS parameter of a service for a user, where the user has never invoked the service in past. For example, here, we want to predict the value of $q_{1,3}$, which is marked by ? symbol.

Table 1. Example of QoS invocation log matrix

$\mathcal{S}$						
$\mathcal{U}$	s_1	s_2	s_3	s_4	s_5	s_6
u_1	0.25	0.3	0 ?	0.301	0	0.01
u_2	0.25	0.33	0.32	0.322	0.1	0
u_3	0.22	0.31	*0.29*	0	0.22	0.01
u_4	0	0	0.31	0.311	0.4	0.15
u_5	0.8	0	0	0.15	0.7	0.99
u_6	0	0	0	0.1	0	0.9

It may be noted that each entry of this matrix essentially represents a QoS invocation log. For example, consider the colored cell, which represents the QoS invocation log $(u_3, s_3, 0.29)$, i.e., the value of the response time of s_3 is 0.29 during the invocation of s_3 by u_3. ■

It may be noted that if a user u_i has never invoked a service s_j, the corresponding entry in the QoS invocation log is $(u_i, s_j, 0)$. In other words, if $\mathcal{Q}(i, j) = 0$, this implies the user u_i has never invoked the service s_j. We now formulate our problem of QoS prediction. We are given the following:

- A set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$.
- A set of web services $\mathcal{S} = \{s_1, s_2, \ldots, s_k\}$.
- For each user u_i, a set of invoked services $\mathcal{S}^i \subseteq \mathcal{S}$.
- For each service s_i, a set of users that invoked s_i, $\mathcal{U}^i \subseteq \mathcal{U}$.
- The QoS invocation log matrix $\mathcal{Q}$ for a given QoS parameter q.
- A target user u_x and a target web service s_y.

The objective of this problem is to predict the value of $q_{x,y}$. In the next section, we demonstrate our solution methodology in detail.

4 Detailed Methodology

Figure 1 illustrates the framework proposed in this paper. Our framework consists of 4 basic modules: (a) a user-intensive filtering module, (b) a service-intensive filtering module, (c) a module for combining the results obtained from the previous steps and (d) a neural network based regression module. Each of the user-intensive and the service-intensive filtering modules again consist of two submodules. Given a target user u_x and a target service s_y, in user intensive module, we first generate a set of users similar to u_x, say $USIM(u_x)$. In the next stage, we find a set of services similar to s_y on $USIM(u_x)$, say $SSIM(u_x, s_y)$. Similarly, in the service-intensive filtering module, we first generate a set of services similar to s_y, say $SSIM(s_y)$, followed by a set of users similar to u_x on $SSIM(s_y)$, say $USIM(s_y, u_x)$. Once we generate, $USIM(u_x)$, $SSIM(u_x, s_y)$, $SSIM(s_y)$ and $USIM(s_y, u_x)$, in our third module, we combine all of them to generate our final user-service QoS invocation log matrix $\mathcal{Q}_{SIM}$. In the final module, we employ a neural network based regression method on $\mathcal{Q}_{SIM}$ to predict the value of $q_{x,y}$. In the following subsections, we discuss each of these modules.

4.1 User-Intensive Filtering

This is the first module of our framework. In this module, we first find a set of users similar to the target user and then find a set of services similar to the target service on the previously computed user-set. We now discuss these two steps in detail.

Find Similar Users. Given a target user u_x, the objective of this step is to find a set of users similar to u_x. Since we do not have any contextual information about a user, the similarity between two users u_i and u_j is calculated from their service-invocation profiles. The key factors that are responsible for measuring the similarity between two users are enlisted below:

(i) The set of web services invoked by either the user u_i or the user u_j, i.e., $(\mathcal{S}^i \cup \mathcal{S}^j)$.
(ii) The set of common services invoked by the user u_i and the user u_j, i.e., $(\mathcal{S}^i \cap \mathcal{S}^j)$.

(iii) The correlation among the QoS values of the services in $(\mathcal{S}^i \cap \mathcal{S}^j)$.

Cosine similarity measure [3] is one such measure which takes all the above factors into account. We now define cosine similarity between two users.

Definition 3 (User Cosine Similarity $SIM_{CS}(u_i, u_j)$). *The cosine similarity between two users u_i and u_j is defined as follows:*

$$SIM_{CS}(u_i, u_j) = \frac{\sum\limits_{s_k \in \mathcal{S}^{i,j}} q_{i,k} \; q_{j,k}}{\sqrt{\sum\limits_{s_k \in \mathcal{S}^i} q_{i,k}^2} \sqrt{\sum\limits_{s_k \in \mathcal{S}^j} q_{j,k}^2}} \tag{1}$$

Where $\mathcal{S}^{i,j} = \mathcal{S}^i \cap \mathcal{S}^j$. ■

It may be noted that the numerator of the above expression is calculated on the set of common services invoked by u_i and u_j, while the denominator is calculated on the individual service invocation profiles of u_i and u_j. The overall expression essentially measures the QoS similarity between two users. Therefore, altogether the cosine similarity measure takes care of all the factors discussed above to compute the similarity between two users.

Given a target user u_x, we now discuss our algorithm to find the set of users similar to u_x. Algorithms 1 and 2 demonstrate our method of finding the similar users.

Algorithm 1. Find Similar Set of Users

```
Input = 𝒰, 𝒮, 𝒬, u_x
Output = USIM(u_x)
for each u_i and u_j ∈ 𝒰 do
    Calculate SIM_CS(u_i, u_j) and store it in a matrix called CosineUser(i, j);
end for
USIM(u_x) = ClusterUsers(CosineUser, t);
return USIM(u_x);
```

In the first step of Algorithm 1, we compute the similarity between each pair of users u_i and u_j in $\mathcal{U}$ using cosine similarity measure as defined in Definition 3. It may be noted, the above definition is commutative, i.e., $SIM_{CS}(u_i, u_j) = SIM_{CS}(u_j, u_i)$. We then perform a clustering to find the set of users similar to u_x. Our proposed clustering algorithm, i.e., Algorithm 2, is a variant of the classical DBSCAN algorithm [8]. The clustering method takes a threshold parameter t as an input. This threshold is a tunable parameter, which is used to decide whether two users are similar. If the similarity measure between $u_i \in \mathcal{U}$ and u_x is more than t, we consider them as similar users and add u_i in $USIM(u_x)$. Here, $USIM(u_x)$ represents the set of users similar to u_x. The transitive similarity between users is also considered in this algorithm. If a user u_i is similar to u_x and another user u_j is similar to u_i, we then add u_j to $USIM(u_x)$, since u_j is transitively similar to u_x. The main motivation behind considering the transitive similarity between users is as follows. The similarity between two users u_i and

u_j is highly dependent on the set of common services they invoked. If u_i and u_j do not invoke any common service, the similarity measure between u_i and u_j becomes 0. However, it may so happen u_j is not similar to u_x, because of less number of common service invocations. Again u_j is highly similar to u_k, which is similar to u_x. In that case, we should consider u_j as well.

Algorithm 2. ClusterUsers

```
Input = CosineUser, t
Output = USIM(u_x)
Add u_x in USIM(u_x);
repeat
    for each new u_i ∈ USIM(u_x) not considered earlier do
        for each u_j ∈ U do
            if USIM_CS(u_i, u_j) ≥ t then
                Add u_j in USIM(u_x), if not already added;
            end if
        end for
    end for
until no new user is added in USIM(u_x);
return USIM(u_x);
```

Example 2 Consider Example 1, where we want to predict the value of $q_{1,3}$. Table 2 shows cosine similarities between each pair of users in $\mathcal{U}$.

Table 2. Example of finding similar users in user-intensive filtering

$\mathcal{U}$						
$\mathcal{U}$	u_1	u_2	u_3	u_4	u_5	u_6
u_1	1	0.84	0.578	0.31	0.35	0.08
u_2		1	0.83	0.62	0.35	0.06
u_3			1	0.56	0.44	0.02
u_4				1	0.53	0.3
u_5					1	0.69
u_6						1

Consider the value of $t = 0.6$. Initially, $USIM(u_1)$ contains only u_1. Using the clustering algorithm discussed above, u_2 is added in $USIM(u_1)$, since $SIM_{CS}(u_1, u_2) = 0.84 > 0.6$. The similarity between u_2 and other users are checked further. Depending on the similarity measures, u_3 and u_4 are added further in $USIM(u_1)$. Therefore, $USIM(u_1) = \{u_1, u_2, u_3, u_4\}$. ■

In the next step of user-intensive filtering, we deal with $USIM(u_x)$ instead of $\mathcal{U}$, where $USIM(u_x) \subseteq \mathcal{U}$. Similarly, instead of dealing with the entire QoS invocation log matrix, we now consider $\mathcal{Q}_u$. $\mathcal{Q}_u$ is a sub-matrix of $\mathcal{Q}$, containing the rows for the users in $USIM(u_x)$.

Find Similar Services. This is the second step of the user-intensive filtering module. Given a target service s_y, the objective of this step is to remove the set of services dissimilar to s_y. The similarity between two services s_i and s_j can be inferred from the following information:

1. The set of common users who invoked s_i and s_j, i.e., $(\mathcal{U}^i \cap \mathcal{U}^j)$.
2. The correlation among the QoS values of s_i and s_j when invoked by the users in $(\mathcal{U}^i \cap \mathcal{U}^j)$.

We use Pearson Correlation Coefficient (PCC) [25] to measure the similarity between the services, since it takes care of all the above factors. We now define PCC similarity below:

Definition 4 (Service PCC Similarity $SIM_{PS}(s_i, s_j)$). *The PCC similarity between two services s_i and s_j is defined as follows:*

$$SIM_{PS}(s_i, s_j) = \frac{\sum\limits_{u_k \in \mathcal{U}^{i,j}} (q_{k,i} - \bar{q_i})(q_{k,j} - \bar{q_j})}{\sqrt{\sum\limits_{u_k \in \mathcal{U}^{i,j}} (q_{k,i} - \bar{q_i})^2} \sqrt{\sum\limits_{u_k \in \mathcal{U}^{i,j}} (q_{k,j} - \bar{q_j})^2}} \tag{2}$$

where $\mathcal{U}^{i,j} = \mathcal{U}^i \cap \mathcal{U}^j$; $\bar{q_i} = \frac{1}{|USIM(u_x)|} \sum\limits_{u_k \in USIM(u_x)} q_{k,i}$. ■

It may be noted, the above definition is commutative, i.e., $SIM_{PS}(s_i, s_j) = SIM_{PS}(s_j, s_i)$.

We now use the same clustering technique as discussed above to find the set of services similar to s_y on the basis of $\mathcal{Q}_u$. The clustering algorithm generates $SSIM(u_x, s_y)$ as output, where $SSIM(u_x, s_y)$ represents the set of services similar to s_y. It may be noted that after this step, we have to deal with $SSIM(u_x, s_y)$ instead of $\mathcal{S}$. Accordingly we change the QoS invocation log matrix. We now consider $\mathcal{Q}_{us}$ instead of $\mathcal{Q}_u$. $\mathcal{Q}_{us}$ is a sub-matrix of $\mathcal{Q}_u$, containing the columns corresponding to the services in $SSIM(u_x, s_y)$. It may be noted, the size of $\mathcal{Q}_{us}$ is $|USIM(u_x)| \times |SSIM(u_x, s_y)|$.

4.2 Service-Intensive Filtering

This is the second module of our framework. In this step, we first find a set of services similar to the target service and then find a set of users similar to the target users on the previously calculated service-set. This method is philosophically similar to the user-intensive filtering method. Below, we discuss the steps of this method briefly.

Find Similar Services. Given a target service s_y, the aim of this step is to find a set of services similar to s_y. Since we do not have any contextual information about a web service, the similarity between two services s_i and s_j is measured from their user-service invocation profiles. As in the case of the user-intensive filtering method, we use the cosine similarity measure [3] to calculate the similarity between two services. We now define cosine similarity between two services as follows.

Definition 5 (Service Cosine Similarity $SIM_{CS}(s_i, s_j)$). *The cosine similarity between two services s_i and s_j is defined as follows:*

$$SIM_{CS}(s_i, s_j) = \frac{\sum_{u_k \in \mathcal{U}^{i,j}} q_{k,i}\ q_{k,j}}{\sqrt{\sum_{u_k \in \mathcal{U}^i} q_{k,i}^2}\sqrt{\sum_{u_k \in \mathcal{U}^j} q_{k,j}^2}} \tag{3}$$

where $\mathcal{U}^{i,j} = \mathcal{U}^i \cap \mathcal{U}^j$. ■

Once we calculate the cosine similarity between each pair of services in $\mathcal{S}$, we use the same clustering technique as discussed in Subsection 4.1 to find the set of services similar to s_y. The clustering algorithm returns $SSIM(s_y)$ as output, which is used in the next step of the service-intensive filtering method. It may be noted that $SSIM(s_y) \subseteq \mathcal{S}$ represents the set of services similar to s_y. Like earlier, we change the QoS invocation log matrix as well. Instead of considering the entire QoS invocation log matrix $\mathcal{Q}$, we now consider $\mathcal{Q}_s$. It may be noted, $\mathcal{Q}_s$ is a sub-matrix of $\mathcal{Q}$, containing the columns corresponding to the services in $SSIM(s_y)$.

Find Similar Users. Given a target user u_x, the objective of this step is to remove the set of users dissimilar to u_x. As in user-intensive filtering, we use Pearson Correlation Coefficient (PCC) [25] to measure the similarity between two users. We now define PCC similarity measure between two users as follows:

Definition 6 (User PCC Similarity $SIM_{PS}(u_i, u_j)$). *The PCC similarity between two users u_i and u_j is defined as follows:*

$$SIM_{PS}(u_i, u_j) = \frac{\sum_{s_k \in \mathcal{S}^{i,j}} (q_{i,k} - \bar{q_i})(q_{j,k} - \bar{q_j})}{\sqrt{\sum_{s_k \in \mathcal{S}^{i,j}} (q_{i,k} - \bar{q_i})^2}\sqrt{\sum_{s_k \in \mathcal{S}^{i,j}} (q_{j,k} - \bar{q_j})^2}} \tag{4}$$

where $\mathcal{S}^{i,j} = \mathcal{S}^i \cap \mathcal{S}^j$ and $\bar{q_i} = \frac{1}{|SSIM(s_y)|} \sum_{u_j \in SSIM(s_y)} q_{i,j}$. ■

The remaining procedure to find the set of users similar to u_x on the basis of $\mathcal{Q}_s$ is same as earlier. The clustering algorithm returns $USIM(s_y, u_x)$ as output, where $USIM(s_y, u_x)$ represents the set of users similar to u_x. It may be noted that after this step, we have to deal with $USIM(s_y, u_x)$ instead of $\mathcal{U}$. Accordingly we change the QoS invocation log matrix. We now consider $\mathcal{Q}_{su}$ instead of $\mathcal{Q}_s$. $\mathcal{Q}_{su}$ is a sub-matrix of $\mathcal{Q}_s$, containing the rows for the users in $USIM(s_y, u_x)$. It may be noted, the size of $\mathcal{Q}_{su}$ is $|USIM(s_y, u_x)| \times |SSIM(s_y)|$.

4.3 Find Similar Set of Users on a Similar Set of Services

The objective of the third module of our framework is to combine the outputs of the user-intensive and service-intensive filtering methods. We take the intersection of the outputs to generate the final result. Consider $SIM(u_x)$ and $SIM(s_y)$ represent the final set of similar users and the final set of similar services respectively. These two sets are calculated as follows:

$$SIM(u_x) = USIM(u_x) \cap USIM(s_y, u_x) \tag{5}$$

$$SIM(s_y) = SSIM(u_x, s_y) \cap SSIM(s_y) \tag{6}$$

Finally, we consider the QoS invocation log matrix as $\mathcal{Q}_{SIM}$, which consists of the rows and columns corresponding to the users in $SIM(u_x)$ and the services in $SIM(s_y)$ respectively.

4.4 Prediction Using Neural Network Based Regression

This is the final module of our framework. Once we obtain the set of similar users $SIM(u_x)$ and the set of similar services $SIM(s_y)$, we employ a neural network based regression module [1] to predict the QoS value of the target service for the target user. Before feeding our data into the neural network, we preprocess the data. In the preprocessing step, we substitute all the 0 entries in $\mathcal{Q}_{SIM}$ by the corresponding column average, except the position that is going to be predicted. The main intuition behind this preprocessing step is as follows. Firstly, $\mathcal{Q}_{SIM}(i, j) = 0$ implies that the user u_i has never invoked the service s_j. Therefore, the 0 entry in $\mathcal{Q}_{SIM}$ does not actually depict the true value of $\mathcal{Q}_{SIM}(i, j)$. Secondly, the column average presents the average QoS values of s_j across all users in $SIM(u_x)$. Therefore, the average value is a better representative value than 0 for $\mathcal{Q}_{SIM}(i, j)$. The modified QoS log matrix is represented by $\mathcal{Q}'_{SIM}(i, j)$.

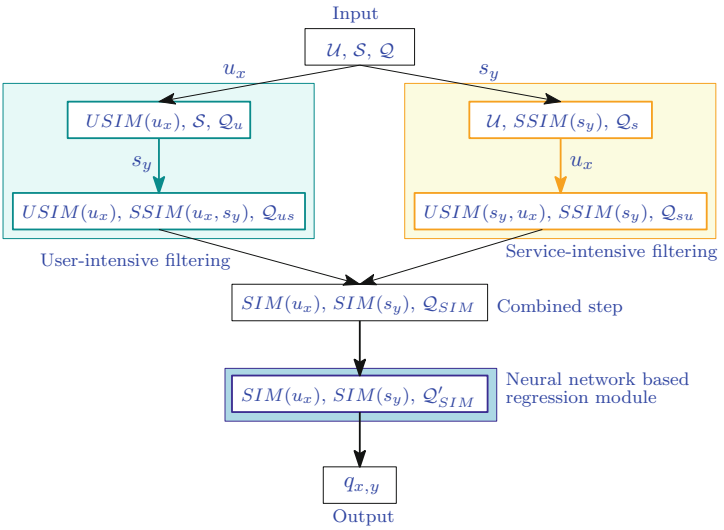

Fig. 2. Data flow in our framework

Finally, $\mathcal{Q}'_{SIM}$ is fed into the neural network. We train the neural network with the service invocation profiles of the following users: $SIM(u_x) \backslash \{u_x\}$. It may

be noted that each training data corresponds to the service invocation profile of a specific user. For each training data, the input layer of the neural network consists of the QoS values of the services in $SIM(s_y) \setminus \{s_y\}$, and the output is the QoS value of s_y for the specific user. The objective is now to obtain the QoS value of s_y for u_x, given the service invocation profile (i.e., the QoS values of the services in $SIM(s_y) \setminus \{s_y\}$) of u_x as input. Figure 2 shows the data flow in our framework.

We now describe the neural network-based regression module [7] used in this paper. We use a linear regression to predict the missing QoS value, i.e., estimating Y, given X by formulating the linear relation between X and Y, as follows, $Y = wX + \beta$. To fit the linear regression line among data points, the weight vector w and bias β are tuned using a neural network architecture [6]. Here, we employ a feed-forward neural network with back propagation, where the weight values are fed forward, and the errors are calculated and propagated back. We use the *traingdx* as training function, since it combines the adaptive learning rate with gradient descent momentum training. *Learngdm* is employed as an adaptive learning function. The *Mean Squared Error (MSE)* measures the performance of the network to assess the quality of the net. *Hyperbolic tangent sigmoid* is used as the transfer function. The experimental setup of this neural network-based regression module is further discussed in Sect. 5.4.

5 Experimental Results

In this section, we demonstrate the experimental results obtained by our framework. We have implemented our framework in MATLAB R2018b. All experiments were performed on a system with the following configuration: Intel Core i7-7600U CPU @ 2.8 GHz with 16 GB DDR4 RAM.

5.1 DataSets

We use the WS-DREAM [24] dataset to analyze the performance of our approach. The dataset comprises of 5,825 web services across 73 countries and 339 web service users across 30 countries. The dataset contains 2 QoS parameters response time and throughput. For each QoS parameter, a matrix with dimension 339×5825 is given. We use the response time matrix to validate our approach.

Training and Testing DataSet. We divide the dataset into two parts: training set and testing set. We use a given parameter $d(0 \leq d \leq 1)$, called density, to obtain the training set. The density is used to denote the proportion of the QoS invocation logs used as the training dataset. For example, if the total number of QoS invocation logs is x and d is the density, the size of the training set then equals to $x \times d$, which is lesser than x. The remaining QoS invocation logs, i.e., $x \times (1 - d)$, are used as the testing dataset.

Each experiment is performed 5 times for each density value. Finally, the average results are calculated and shown in this paper.

5.2 Comparative Methods

We compare our approach with the following approaches from the literature:

- UPCC [3]: This method employs a user-intensive collaborative filtering approach for QoS prediction.
- IPCC [14]: This approach employs service-intensive collaborative filtering for QoS prediction.
- WSRec [22]: This method combines UPCC and IPCC.
- NRCF [16]: This method employs classical collaborative filtering to improve the prediction accuracy.
- RACF [21]: Ratio based similarity (RBS) is used in this work and the result is calculated by the similar users or similar services.
- RECF [25]: Reinforced collaborative filtering approach is used in this work to improve the prediction accuracy. In this method, both user-based and service-based similarity information are integrated into a singleton collaborative filtering.
- MF [11]: Matrix factorization based approach is used here for prediction.
- HDOP [18]: This method uses multi-linear-algebra based concepts of tensor for QoS value prediction. Tensor decomposition and reconstruction optimization algorithms are used to predict QoS value.

As discussed earlier in this paper, we propose a collaborative filtering approach followed by the neural network-based regression model (CNR). To show the necessity of each step of our approach, we further compare our method with the following approaches.

- NR: In this approach, we only consider the neural network-based regression model, without using any collaborative filtering method.
- CR: In this approach, we use the same collaborative filtering method as demonstrated in this paper. However, instead of using a neural network-based linear regression model, a simple linear regression module is used here to predict the QoS value.
- UCNR: In this approach, we use the user-intensive collaborative filtering method along with the neural network-based regression model.
- SCNR: In this approach, we use the service-intensive collaborative filtering method along with the neural network-based regression model.
- CNRWoV: This approach is same as CNR. The only difference here, we do not substitute the 0 entries in $\mathcal{Q}_{SIM}$ by the corresponding column average.
- CNRCC: In this approach, we use cosine similarity measure to find similar users and services for both user-intensive and service-intensive filtering methods.

5.3 Comparison Metric

We use *Mean Absolute Error (MAE)* [25] to measure the prediction error in our experiment. It may be noted that lower the value of MAE, better is the prediction accuracy.

Definition 7 (Mean Absolute Error (MAE)). *MAE is defined as follows:*

$$MAE = \frac{\sum_{q_{i,j} \in TD} |q_{i,j} - \hat{q}_{i,j}|}{|TD|}$$

where, $q_{i,j}$ represents the ground truth QoS value of the j^{th} service for the i^{th} user in the testing dataset TD. $\hat{q}_{i,j}$ represents the predicted QoS value for the same. ■

5.4 Configuration of Our Experiment

To generate the set of similar users and services, empirically we chose the user-threshold value between 0.5 to 0.6 and service-threshold value between 0.4 to 0.5 for our clustering methods. Later in this section, we show how the change of the threshold value impacts on the prediction quality.

For the neural network-based regression model, we used the following configuration in our experiment. We considered 2 hidden layers in the neural network. We varied the number of neurons in each hidden layer within the range [4, 128]. Finally, we obtained the best results for 16 neurons in the first hidden layer and 8 neurons in the second hidden layer. Among the hyper-parameters, the learning rate was set to 0.01 with momentum 0.9. The training was performed up to 1000 epochs or up to minimum gradient of 10^{-5}.

Table 3. Comparative study (MAE) on different prediction methods

Density	UPCC	IPCC	WSRec	NRCF	RACF	RECF	MF	HDOP	**CNR**
0.10	0.6063	0.7000	0.6394	0.5312	0.4937	0.4332	0.5103	0.3076	**0.2597**
0.20	0.5379	0.5351	0.5024	0.4607	0.4208	0.3946	0.4981	0.2276	**0.1711**
0.30	0.5084	0.4783	0.4571	0.4296	0.3997	0.3789	0.4632	0.1841	**0.0968**

We present partial comparative study from Fig. 3(a) due to space constraint

5.5 Analysis of Results

Figure 3(a) and (b) show a comparative study for QoS prediction by different approaches. Table 3 shows partial comparative results of Fig. 3(a) in a more quantitative way. From our experimental results, we have the following observations:

(i) It is evident from Table 3 and Fig. 3(a) that among all the approaches, our proposed approach (CNR) produces the best result in terms of the prediction accuracy, as CNR has the lowest MAE value among all the approaches for each density value.

(ii) It can also be observed from Table 3 and Fig. 3(a) and (b) that as the density increases, the value of MAE decreases. This is mainly because of the fact that as the density increases, the number of QoS invocation logs in the training dataset increases and thereby, the prediction accuracy increases.

(iii) Figure 3(b) shows the requirement of each step of our proposal. As is evident from the figure, CNR is better than NR, which explains the requirement of the collaborative filtering approach. CNR is also better than CR, which confirms the importance of the neural network-based regression model. On one side, CNR is better than UCNR, on the other side, CNR is better than SCNR, which indicates the necessity of our combine step. Further, we compare CNR with CNRWoV, which shows the significance of replacing 0 entries in $\mathcal{Q}_{SIM}$ by the corresponding column average.

In CNR, we use the cosine similarity measure followed by PCC (i.e., cosine + PCC). We have, therefore, further experimented our framework with other combinations of similarity measures, such as cosine+cosine, PCC+PCC, PCC+cosine, which did not work well in comparison with the cosine + PCC. In Fig. 3(b), we present only the result of CNRCC (i.e., cosine+cosine), which worked the second best.

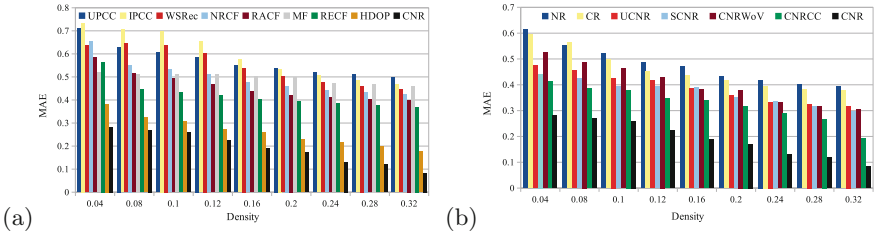

Fig. 3. Comparative study on different prediction methods

5.6 Impact of the Tunable Parameters on Our Experiment

In this subsection, we discuss the impact of the tunable parameters on the results obtained by our proposed method. We used 4 tunable threshold parameters in our experiments, i.e., a threshold value required to cluster the users and services in the user-intensive and service-intensive filtering steps. However, we used the same threshold value to cluster the users (services) in both the user-intensive and service-intensive filtering steps.

Figure 4(a) shows the variation of MAE (along the y-axis) with respect to the threshold (along the x-axis) required to cluster the services for a constant threshold (shown as legends in the graph) required for user clustering. Similarly, Fig. 4(b) shows the variation of MAE (along the y-axis) with respect to the the threshold (along the x-axis) required to cluster the users for a fixed threshold (shown as legends in the graph) required for service clustering.
From Fig. 4 (a) and (b), we have the following observations:

(i) As evident from both the figures, for the threshold value between 0.4 to 0.6, we obtain better results in terms of prediction accuracy.

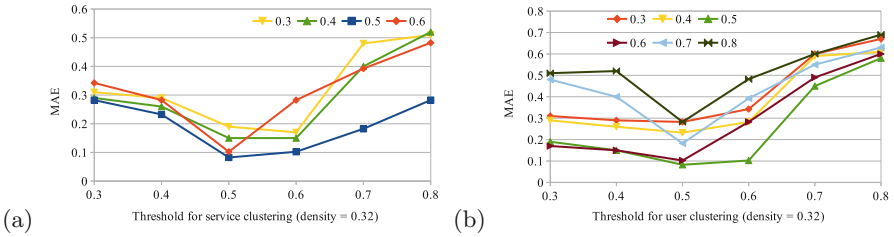

Fig. 4. Variation of MAE across the threshold used for (a) user clustering, (b) service clustering

(ii) For a very low value of the threshold, we may end up having the entire QoS logs in the training dataset. In this case, we obtain the same results as NR method.

(iii) For a very high threshold value, we end up having very less number of similar users and similar services which are insufficient to train the neural network-based regression model and thereby the prediction accuracy decreases.

In summary, as evident from our experiment, our proposed method outperformed the major state-of-the-art methods in terms of prediction accuracy.

6 Conclusion

In this work, we propose a method to predict the value of a given QoS parameter of a target web service for a target user. We leverage the collaborative filtering approach along with the regression method. We conducted our experiments on the WS-DREAM dataset. The experimental results show that our method is more efficient in terms of prediction accuracy than the past approaches. However, in this paper, we do not consider the fact that QoS parameters vary across time as well. Even for a single user, the QoS value of a service can be different across time. We wish to take up this task of QoS value prediction in a dynamic environment going ahead.

References

1. Adamczak, R., et al.: Accurate prediction of solvent accessibility using neural networks-based regression. Proteins Struct. Funct. Bioinform. **56**(4), 753–767 (2004)
2. Amin, A., et al.: An approach to forecasting qos attributes of web services based on arima and garch models. In: ICWS, pp. 74–81. IEEE (2012)
3. Breese, J.S., et al.: Empirical analysis of predictive algorithms for collaborative filtering. In: Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann Publishers Inc. (1998)
4. Chattopadhyay, S., et al.: A framework for top service subscription recommendations for service assemblers. In: IEEE SCC, pp. 332–339 (2016)

5. Chen, X., et al.: Personalized qos-aware web service recommendation and visualization. IEEE TSC **6**(1), 35–47 (2013)
6. Daniel, G.: Principles of Artificial Neural Networks, vol. 7. World Scientific, Singapore (2013)
7. Demuth, H., Beale, M.: Neural Network Toolbox, vol. 4. The MathWorks Inc., Boston (2004)
8. Ester, M., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 96, pp. 226–231 (1996)
9. Li, S., Wen, J., Luo, F., Ranzi, G.: Time-aware QoS prediction for cloud service recommendation based on matrix factorization. IEEE Access **6**, 77716–77724 (2018)
10. Li, S., et al.: From reputation perspective: a hybrid matrix factorization for QoS prediction in location-aware mobile service recommendation system. Mob. Inf. Syst. **2019**, 8950508:1–8950508:12 (2019)
11. Lo, W., et al.: An extended matrix factorization approach for qos prediction in service selection. In: IEEE SCC, pp. 162–169. IEEE (2012)
12. Ma, Y., et al.: Predicting QoS values via multi-dimensional QoS data for web service recommendations. In: ICWS, pp. 249–256. IEEE (2015)
13. Qi, K., et al.: Personalized QoS prediction via matrix factorization integrated with neighborhood information. In: SCC, pp. 186–193. IEEE (2015)
14. Sarwar, B.M., et al.: Item-based collaborative filtering recommendation algorithms. WWW **1**, 285–295 (2001)
15. Shao, L., et al.: Personalized qos prediction for web services via collaborative filtering. In: IEEE ICWS, pp. 439–446 (2007)
16. Sun, H., et al.: Personalized web service recommendation via normal recovery collaborative filtering. IEEE TSC **6**(4), 573–579 (2013)
17. Tang, M., et al.: Location-aware collaborative filtering for QoS-based service recommendation. In: ICWS, pp. 202–209. IEEE (2012)
18. Wang, S., et al.: Multi-dimensional QoS prediction for service recommendations. IEEE TSC **12**, 47–57 (2016)
19. Wu, C., Qiu, W., et al.: Time-aware and sparsity-tolerant QoS prediction based on collaborative filtering. In: IEEE ICWS, pp. 637–640 (2016)
20. Wu, H., et al.: Collaborative QoS prediction with context-sensitive matrix factorization. Future Gener. Comp. Syst. **82**, 669–678 (2018)
21. Wu, X., et al.: Collaborative filtering service recommendation based on a novel similarity computation method. IEEE TSC **10**(3), 352–365 (2017)
22. Zheng, Z., et al.: QoS-aware web service recommendation by collaborative filtering. IEEE TSC **4**(2), 140–152 (2011)
23. Zheng, Z., et al.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE TSC **6**(3), 289–299 (2013)
24. Zheng, Z., et al.: Investigating qos of real-world web services. IEEE TSC **7**(1), 32–39 (2014)
25. Zou, G., Jiang, M., Niu, S., Wu, H., Pang, S., Gan, Y.: QoS-aware web service recommendation with reinforced collaborative filtering. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds.) ICSOC 2018. LNCS, vol. 11236, pp. 430–445. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03596-9_31