A Missing QoS Prediction Approach via Time-aware Collaborative Filtering

Endong Tong, Wenjia Niu and Jiqiang Liu

Abstract—Quality of Service (QoS) guarantee is an important issue in building service-oriented applications. Generally, some *QoS* values of a service are unknown to its users who have never invoked the service before. Fortunately, collaborative filtering (*CF*)-based methods are proved feasible for missing *QoS* prediction and have been widely used. However, these methods seldom took the temporal factors into consideration. Indeed, historical *QoS* values contain more information about user (or service) similarity. Furthermore, as the application environment is dynamic, obtained *QoS* values usually have short timeliness. Hence, using outdated *QoS* values will largely decrease the prediction accuracy. In order to resolve this issue, we proposed a time-aware collaborative filtering approach. Firstly, we proposed a *QoS* model to filter out outdated *QoS* values, and divided the obtained *QoS* values into several time slices. Then, we computed the average value of historical *QoS* as temporal *QoS* forecast. In addition, by introducing time-aware similarity computation mechanism, we succeeded to select real similar neighbor users (or services) and further predict the *CF*-based *QoS* based on *CF* technology. Finally, we can predict the final missing *QoS* by combining temporal *QoS* forecast and *CF*-based *QoS* prediction. Experiment results show that our approach can receive better prediction precision.

Index Terms—QoS prediction, time-aware, collaborative filtering, service recommendation

1 Introduction

With the development of network technology and the strong needs of end users, web services have enjoyed great boosts. More and more web services with the same or similar function have been provided in networks. Hence, how to select the most suitable services for a user has become a crucial and challenging task.

Although many web services have the same or similar function, they usually have different Quality of Service (*QoS*) due to different service providers and network environment. Meanwhile, users generally have *QoS* requirements when selecting web services. Hence, service selection and composition must consider both functional and non-functional attributes of services [1]. Recently, *QoS* guaranteed web service discovery and selection has aroused a great deal of research interests [2], [3], [4].

However, *QoS* of services includes a set of non-functional properties [5], such as response time, reliability, price, etc., which characterize various aspects of service quality. Some *QoS* properties are user-independent, such as service price. While some other *QoS* properties are user-dependent, such as response time. For user-dependent *QoS* properties, real invocations must be done to acquire concrete *QoS* values. Generally, it is neither easy nor practical for a user to get the *QoS* of all candidate web services, due to the following reasons: (1) The number of web services in real applications is very huge. Therefore, it is time-consuming and difficult for a user to make all web service invocations. (2) The application environment becomes more dynamic and vulnerable nowadays. Furthermore, it turns out to be impractical and impossible to acquire *QoS* values all the time. Under the above circumstances, if one user has not invoked one service before, the user-dependent *QoS* values are unknown for this user. In fact, there exists a great number of unknown user-dependent *QoS* values (i.e., missing *QoS* values), which will largely affect the accuracy of *QoS* aware service discovery and selection.

Recently, many methods have been proposed to predict unknown *QoS* values [6]. In missing *QoS* prediction process, historical known *QoS* data is usually too sparse to fit in with traditional time series forecasting models [7]. Collaborative filtering(*CF*) is a method of making predictions about the interests of a user by collecting preference information from many other users. It assumes that if some users all like many of the same things, these users will also concurrently like other things. For example, if a set of users have watched a number of movies and meanwhile they provide same evaluations for these movies. We believe that these users hava high similarity. Hence, we can recommend a movie that these users loved to the one among them who has not watched the movie. In missing *QoS* prediction scene, *CF* is utilized to predict missing *QoS* of a user based on the known *QoS* of his similar users. In the same way, it assumes that if some users have similar *QoS* on many of the same services, these users will also have similar *QoS* on other services. Recently, as the most popular and successful recommendation method, *CF* has been widely used in many missing *QoS* prediction work [8], [9].

1.1 Motivation

In this section, we firstly present a scenario to illustrate the motivation of our work. Assume that there exists a *QoS*-based web service recommendation system which consists of service providers and service users besides system itself.

- *Endong Tong, Wenjia Niu, and Jiqiang Liu are with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, P.R. China.*
- *Corresponding authors are Endong Tong and Wenjia Niu. E-mail: {edtong, niuwj}@bjtu.edu.cn*

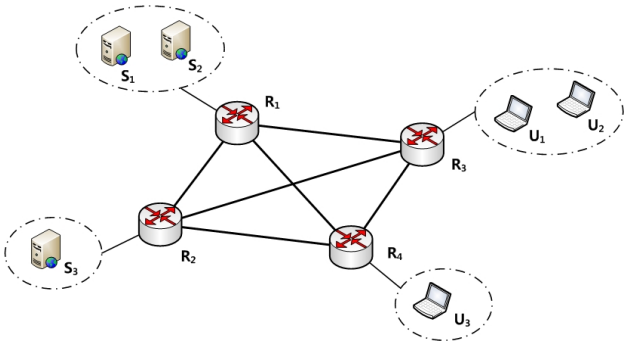


Fig. 1: Motivation Example.

As shown in Fig. 1, there are three users (U_1, U_2 and U_3) and three web services (S_1, S_2 and S_3). Among of them, U_1 and U_2 are in the same autonomous system(AS), while S_1 and S_2 are also in the same AS. Hence, U_1 and U_2 are likely to observe similar *QoS*, such as response time, on web services (e.g., S_1, S_2 and S_3).

In this scenario, users will invoke services from time to time. Generally, surrounding network environment will affect the associated *QoS*. Hence, according to the network environment, we divide the historical *QoS* into several time slices. In intra-slice, the network status is almost the same. While, in inter-slices, the network status is very different. Then, the concrete *QoS* (response time in Table. 1) values in different time slices can be seen in Table. 1.

TABLE 1: Temporal QoS Data

		S_1	S_2	S_3
T_1	U_1	160	/	/
	U_2	/	138	/
	U_3	130	/	/
T_2	U_1	130	/	30
	U_2	/	108	32
	U_3	/	80	60
T_3	U_1	100	80	30
	U_2	100	78	32
	U_3	70	50	60

The digital number in the table indicates the user has invoked the service, while "/" means the user did not invoke the service in that time slice. T_1 is the nearest time period, while T_3 is the farthest time period. In T_3, the *QoS* sets of U_1, U_2 and U_3 are as follows,

$\mathcal{R}_{U_1} = \{100, 80, 30\}$
$\mathcal{R}_{U_2} = \{100, 78, 32\}$
$\mathcal{R}_{U_3} = \{70, 50, 60\}$.

The calculated similarity between U_1 and U_2 is 0.999, and the similarity between U_1 and U_3 is 0.277.

In T_2, there exists a congestion at router R_1, which increases the response time by 30. Hence, the response time observed by users on service S_1 or S_2 will eventually increased by 30. In T_1, the above congestion at router R_1 continues, which increases the response time by 60. Then, after T_1, the *QoS* sets of U_1, U_2 and U_3 are as follows,

$\mathcal{R}_{U_1} = \{160, 80, 30\}$
$\mathcal{R}_{U_2} = \{100, 138, 32\}$
$\mathcal{R}_{U_3} = \{130, 80, 60\}$.

Finally, the calculated similarity between U_1 and U_2 is 0.525, and the similarity between U_1 and U_3 is 0.994.

As we know, U_1 and U_2 are in the same *AS*, they should have higher similarity. However, as traditional algorithms ignore the temporal characteristic of *QoS*, the obtained similarity between U_1 and U_2 is largely lower than the real one. At the same time, U_1 and U_3, which are very different with each other actually, have higher similarity according to existing similarity computation algorithms.

In addition, we also perform an analysis on *WS-DREAM Dataset 3* [10], which describes *QoS* records of service invocations over 64 consecutive time slices. Generally, not all the services have been invoked by some user at every time slice. Hence, there exist many missing *QoS*. For every time slice, we calculate the similarity of any two users. As traditional algorithms ignore the temporal characteristic of *QoS*, we find that some users' similarity fluctuates largely according to the time slice. This means that at some time slices users with high similarity actually will be mistook for dissimilar ones, or users with low similarity actually will be mistook for similar ones.

In real applications, service invocations from different users are discretional and independent. Hence, for one specific service, *QoS* values from various users are usually calculated at different points in time. As the surrounding environment is dynamic and service *QoS* is highly depend on services' circumstances, service invocations at different points in time may have totally different *QoS* values. For example, both user u and user v invoked the same service s, and the corresponding latest *response time* was $r(u, s)$ and $r(v, s)$ respectively. However, $r(u, s)$ was calculated just a few minutes ago, while $r(v, s)$ was calculated one day ago. During the past day, the application environment may have changed a lot. It is less persuasive to use $r(u, s)$ and $r(v, s)$ to get the similarity of user u and user v.

Most previous *CF*-based missing *QoS* prediction approaches just use the latest or average user-service *QoS* value to find similar users, which fail to consider the temporal factors. However, the surrounding network environment is dynamic. Under this circumstance, two users who are not really similar, may happen to have similar *QoS* experiences on a few services. As a consequence, these two users will be mistaken as similar users.

1.2 Research Issues

From the above motivation expression, we come to the conclusion that the temporal characteristic of *QoS* will largely affect the similarity evaluation, and further decrease the accuracy of missing *QoS* prediction. Hence, in order to make the best service discovery and service selection, historical multidimensional *QoS* values must be exploited as much as possible when predicting missing *QoS* values [11].

In this paper, we aim to realize accurate *QoS* prediction by addressing the following key issues:

- Time-aware user-service *QoS* model. Generally, users will invoke one service more than one time. As application environment is dynamic, *QoS* values have the

characteristic of timeliness. Hence, we will develop a *QoS* model, which can represent temporal *QoS* values and facilitate the following missing *QoS* prediction.
- Collaborative filtering based missing *QoS* prediction. Two users who are not really similar, may be taken for similar users once in a while. But, in most cases, they are inevitably considered as irrelevant users. We will make full use of historical temporal *QoS* values, in order to predict missing *QoS* with higher accuracy.

The rest of this paper is organized as follows. In Sec. 2, we give the related work. Then, Sec. 3 proposes the time-aware user-service *QoS* model, followed by collaborative filtering based missing *QoS* prediction mechanism in Sec. 4. Sec. 5 provides experiments that illustrate the benefits of our proposed approach. Finally, Sec. 6 concludes this paper and discusses future research.

2 Preliminaries and Related Work

In order to facilitate the understanding of our time-aware *QoS* prediction approach, this section introduces the basic procedure of *PCC*-based collaborative filtering method and related research work.

2.1 Collaborative Filtering with Pearson Correlation Coefficient

Pearson correlation coefficient (*PCC*) is a measure of the linear correlation between two variables X and Y. It has a value between +1 and -1, where +1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. Now, *PCC* and its enhanced methods have been commonly used in collaborative filtering algorithms. There are two important steps in *PCC*-based collaborative filtering. The first one is to compute the similarity between users or(and) between services. The second one is to predict the missing *QoS* values, according to the known *QoS* values of similar users or(and) similar services.

User-based Collaborative Filtering

Suppose there are user set U and service set S. A user u in U invokes a service s in S, and the corresponding *QoS* is represented as $r(u,s)$. Then the similarity $Sim(u,v)$ between user u and user v can be calculated as below,

$$Sim(u,v)=\frac{\sum_{s\in\widetilde{S}}(r(u,s)-\overline{r(u)})\cdot(r(v,s)-\overline{r(v)})}{\sqrt{\sum_{s\in\widetilde{S}}(r(u,s)-\overline{r(u)})^2}\cdot\sqrt{\sum_{s\in\widetilde{S}}(r(v,s)-\overline{r(v)})^2}} \tag{1}$$

where $Sim(u,v)$ is in the interval of [-1, 1]. $\widetilde{S}=S(u)\cap S(v)$ is the service set that are invoked by both u and v. $\overline{r(u)}$ and $\overline{r(v)}$ represent the average *QoS* values that u and v have perceived from services in $\widetilde{S}$, respectively.

If user u has not invoked service s, the corresponding $r(u,s)$ is missing. However, there exists some other similar users of u whose $r(v,s)$ are already known. Thus, based on these already known *QoS* values, we can predict $r(u,s)$ with high precision. User-based collaborative filtering prediction for $r_{upcc}(u,s)$ is as below,

$$r_{upcc}(u,s)=\overline{r(u)}+\frac{\sum_{v\in SU(u)}(Sim(u,v)\cdot(r(v,s)-\overline{r(v)}))}{\sum_{v\in SU(u)}(Sim(u,v))} \tag{2}$$

where $SU(u)$ denotes the similar user set of user u.

Item-based Collaborative Filtering

Meanwhile, the similarity $Sim(s,f)$ between service s and service f can be calculated as below,

$$Sim(s,f)=\frac{\sum_{u\in\widetilde{U}}(r(u,s)-\overline{r(s)})\cdot(r(u,f)-\overline{r(f)})}{\sqrt{\sum_{u\in\widetilde{U}}(r(u,s)-\overline{r(s)})^2}\cdot\sqrt{\sum_{u\in\widetilde{U}}(r(u,f)-\overline{r(f)})^2}} \tag{3}$$

where $Sim(s,f)$ is in the interval of [-1, 1]. $\widetilde{U}=U(s)\cap U(f)$ is the user set who have invoked both s and f. $\overline{r(s)}$ and $\overline{r(f)}$ represent the average *QoS* value of s and f invoked by users in $\widetilde{U}$, respectively.

If user u has not invoked service s, the corresponding *QoS* $r(u,s)$ is missing. However, there exists some other similar services whose $r(u,f)$ are already known. Thus, based on these already known *QoS* values, we can predict $r(u,s)$. Item(service)-based collaborative filtering prediction for $r_{ipcc}(u,s)$ is as below,

$$r_{ipcc}(u,s)=\overline{r(s)}+\frac{\sum_{f\in SS(s)}(Sim(s,f)\cdot(r(u,f)-\overline{r(f)}))}{\sum_{f\in SS(s)}(Sim(s,f))} \tag{4}$$

where $SS(s)$ denotes the similar service set of service s.

Combination of User-based and Item-based Collaborative Filtering

In order to improve the prediction accuracy, many work have integrated user-based *PCC* and item-based *PCC*. Hence, $r(u,s)$ can be predicted as below,

$$r(u,s)=\lambda\cdot r_{upcc}(u,s)+(1-\lambda)\cdot r_{ipcc}(u,s) \tag{5}$$

where $\lambda\in[0,1]$ determines the ratio how the final result depends on user-based *PCC* and item-based *PCC*.

2.2 Related Work

Many types of methods have been applied for missing *QoS* prediction, such as time series forecasting [12] [13] and collaborative filtering [14], etc. Among of them, collaborative filtering has been widely used as its better performance.

There are two types of collaborative filtering approaches: the model-based approach and the memory-based approach [15], [16]. Matrix factorization [17] is a kind of model-based approach. It factorizes the *QoS* matrix R into a latent user space U and a latent service space S, in such a way that R can be well fitted by the inner product of U and S. After obtaining U and S, a specific missing *QoS* can thus be predicted using the corresponding inner product. Memory-based approach is also named neighborhood-based approach. It is mainly implemented by user-based methods [18], item-based methods [19] and their hybridization [20]. Similarity computation between users or items is an important part in collaborative filtering. In most literatures, memory-based approach calculate the similarity by employing *PCC* [21]. Based on the traditional *CF* approaches, several enhanced methods have been proposed to improve the prediction accuracy. In collaborative filtering

based service selection, similarity computation is an important step. Zheng et al. [8] proposed an enhanced *PCC* similarity computation method, which employ a significance weight to reduce the influence of a small number of similar coinvoked items. Jiang et al. [22] suggested that services with more stable *QoS* from user to user should be given smaller weight in user similarity computation. Wu et al. [23] proposed an improved *CF* method by using data smoothing for the user-service *QoS* matrix.

Several influential factors of services should also be considered in *CF*-based missing *QoS* prediction. Liu et al. [24] indicated that locations of users and services are crucial factors affecting *QoS*. Similar users which are computed by traditional *CF* methods may not be really similar, but happen to have similar *QoS* experiences on a few services. Hence, they incorporate the location information (i.e., autonomous system and country) of users and services into similar neighbor selection. Ma et al. [25] assumed that if two users share a high similarity, their similarity will hardly fluctuate with their future invocations of more numbers of services. They set the new similarity the same with the old similarity to get a quadratic equation. Through solving the quadratic equation, they got two possible predicted *QoS* values. Then, they selected the right one by the aid of a linear regression process.

Many works have taken into account the time-varying characteristic of *QoS*. *ARIMA* is widely used as its lower prediction error [26]. Although *ARIMA* models have good performance, they only forecast the future *QoS* values for each service individually. In addition, the sparse historical *QoS* values are not sufficient to construct the time series forecasting model. A novel similarity-enhanced collaborative filtering (*CF*) approach [27] is developed to address the data sparsity issue, and then *ARIMA* is applied to predict missing *QoS*. In [28], Matrix factorization is applied to extract user-preferences and service-features, respectively. Afterwards, the Long Short Term Memory (*LSTM*) model is leveraged to learn and predict preferences and features. Then, a service recommendation list is generated. However, these research mainly focus on service recommendation that tries to recommend Top-N services to specific users. Zhang et al. [29] formalized missing *QoS* prediction problem as a generalized tensor factorization model and proposed a Non-negative Tensor Factorization (*NTF*) algorithm. Wu et al. [30] computed the average value of historical *QoS* values and combined it with the prediction results from *CF*. However, they ignores the impact of temporal distance of historical data on prediction results. Li et al. [31] proposes a time-aware matrix factorization (*TMF*) model, which uses an adaptive matrix factorization to predict the missing *QoS* values. A temporal smoothing method is then applied to the predicted result to perform the time-varying *QoS* prediction. Zhang et al. [32] develop a service recommendation model based on Recurrent Tensor Factorization (*RTF*). Personalized Gated Recurrent Unit (PGRU) and Generalized Tensor Factorization (*GTF*) simultaneously work on shared embedding dense vectors to predict missing QoS. Chen et al. [33] propose a Separated Time-aware Collaborative Poisson Factorization (*STCPF*) for service recommendation. *STCPF* takes Poisson Factorization as the foundation to model mashup queries and service descriptions separately, and incorporates them with the historical usage data together by using collective matrix factorization. Qi et al. [34] propose a time-aware and privacy-preserving service recommendation approach named SerRec based on Locality-Sensitive Hashing (*LSH*). Tian et al. [35] propose a time-aware service recommendation approach based on users' implicit feedback on services. In *TASR*, three time effects are analyzed and modeled including user bias shifting, Web service bias shifting, and user preference shifting. Fayala et al. [36] uses K-means to exclude the less similar users. Slope One algorithm is also adopted to predict the missing ratings over time. Then, a recommendation algorithm is presented in order to recommend the top-rated services.

Obviously, the *QoS* value is highly related to the invocation context. Existing approaches usually focus on the top service recommendation or the service *QoS* prediction at specific time slice. In addition, some existing approaches do not consider much about the dynamic characteristic of network environment. Hence, based on the historical *QoS* records of the previous time slices, we make the prediction of current missing *QoS* which is crucial to the success of service selection and further service composition. Hence, in this paper, we will try to address the temporal factors and further propose a time-aware collaborative filtering based missing *QoS* prediction method.

3 Time-aware User-Service QoS Model

In this section, we will propose our time-aware *QoS* model. With this *QoS* model, we can further perform the following missing *QoS* prediction.

3.1 Notations and Definitions

We first define some notations which will be employed in the rest of this paper:

- $U = \{u_1, u_2, ..., u_m\}$ is the whole set of users in the service system, where m is the total number of users and $u_i (1 \leq i \leq m)$ represents a user in U.
- $U(s) = \{u_1, u_2, ...\}$ is a set of users who have invoked service s, and u_i represents a user in $U(s)$.
- $S = \{s_1, s_2, ..., s_n\}$ is the whole set of services in the service system, where n is the total number of services, and $s_j (1 \leq j \leq n)$ represents a service in S.
- $S(u) = \{s_1, s_2, ...\}$ is a set of services which have been invoked by user u, and s_j represents a service in $S(u)$.
- $R(u, s) = \{r(u, s) | u \in U, s \in S\}$ is a user-service matrix, where $r(u, s)$ represents the *QoS* value generated during the invocation between user u and service s.

To aid the following representation, we also give some definitions.

Definition 1 (Current User). The user who wants to invoke a service, while its corresponding QoS value is missing, thus he requires a prediction for this unknown *QoS*.

Definition 2 (Current Service). The service that a current user has not invoked but he prepare to do so. This service is a recommendation candidate for current user.

Definition 3 (Active User). If a current user u requires the unknown *QoS* value $r(u, s)$, and $r(v, s)$ is already known, then user v is an active user to user u.

Definition 4 (Active Service). If a user u requires the unknown *QoS* value $r(u, s)$ on a current service s, and $r(u, f)$ is already known, then service f is an active service to service s.

Definition 5 (Similar User). The users whose similarity with the current user is larger than a given threshold.

Definition 6 (Similar Service). The services whose similarity with the current service is larger than a given threshold.

3.2 QoS Model

Generally, users will invoke one service more than one time. *QoS* values could be calculated when each service invocation has finished. Take *response time* for example, we can compute the time period from the beginning of service request to the time when service invocation finished. Every time a service invocation complete, we can get a corresponding value of *response time*. As a result, for each user-service pair, the corresponding *QoS* values will be a temporal *QoS* set. In order to avoid the influence coming from outdated *QoS* values, we will only choose *QoS* values during the nearest T.

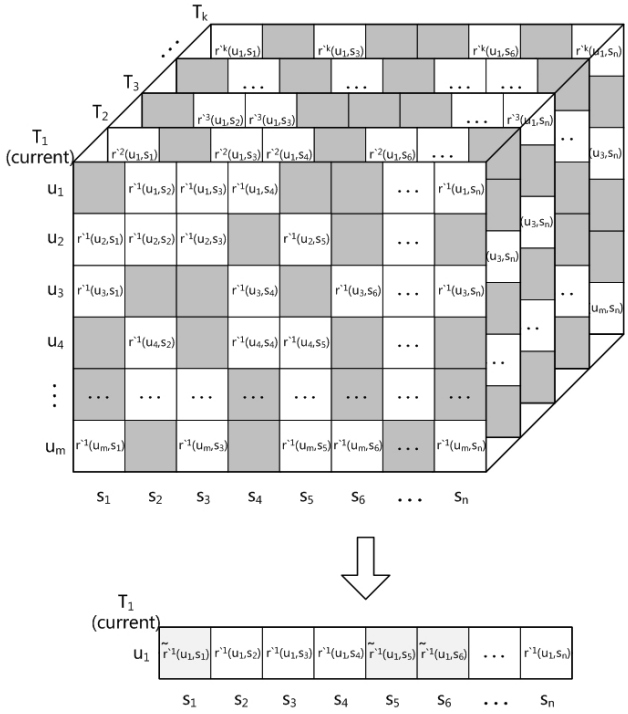


Fig. 2: *QoS* Matrix.

$$R(u, s) = \{r(u, s, t)|u \in U, s \in S, t_{current}-T \leq t \leq t_{current}\} \tag{6}$$

$t_{current}$ represents the time when the missing *QoS* prediction request comes. As the application environment will rarely change during a short time interval, we divided the historical *QoS* values into k time slices (each has a time span of $\frac{T}{k}$). Hence, *QoS* values in the same time slice can be aggregated to represent the user-service relationship at that time. Then, in order to get accurate user-service relationship, for each time slice, we sum up all the *QoS* values in that time slice and then get the average one. In this way, we can get a new temporal *QoS* set $R'(u, s)$ with k items for each user-service pair.

$$R'(u, s) = \{r'^{i}(u, s), i = 1, 2, ..., k\} \tag{7}$$

$$r'^{i}(u, s) = \frac{1}{N_i} \cdot \sum_{t} (r(u, s, t)|t_b \leq t \leq t_u) \tag{8}$$

$r'^{i}(u, s)$ is the average *QoS* value of the i-th time slice, $t_b = t_{current} - \Delta T \cdot i$, $t_u = t_{current} - \Delta T \cdot (i-1)$, $\Delta T = \frac{T}{k}$, N_i is the number of *QoS* values in time span (t_b, t_u).

Hence, the user-service *QoS* values can be represented as a user-service-time *QoS* matrix. Assuming that there are m users, n services and k time intervals, an $m \cdot n \cdot k$ matrix $\mathcal{Q}$ illustrated in Fig. 2 is used to represent the user-service-time relationship. Suppose that user u_1 invoked service s_2 at time T_1, the observed *QoS* value $r'^{1}(u_1, s_2)$ is recorded in the entry (u_1,s_2,T_1).

Generally, there may be no *QoS* values in some time slices. Then, the corresponding $r'^{i}(u, s)$ would be missing, such as $r'^{1}(u_1, s_1)$ and $r'^{2}(u_1, s_2)$ in Fig. 2. As we know, in order to perform the further *QoS*-aware service recommendation, we have to get all the current *QoS* values. Hence, the aim of our research is to predict the missing *QoS* values in the first time slice (i.e., $r'^{1}(u, s)$) based on the overall user-service-time *QoS* matrix.

4 Collaborative Filtering based Missing QoS Prediction

By assigning T with a relative small value, *QoS* prediction approaches can remove outdated *QoS* values, which will largely improve the prediction accuracy. However, the value selection of T will be a tough work. As we know, larger T will bring more outdated *QoS* values into prediction process, while small T will reduce the number of *QoS* values used for prediction.

Hence, in order to resolve the above problem, we proposed our time-aware missing *QoS* prediction approach. There are two issues should be considered. As the application environment is usually dynamic, obtained *QoS* values will have short timeliness. In addition, *QoS* values related to web services is objective. They are determined as a result of some objective factors related to users, such as user location and network bandwidth. Then, based on the above two issues, we have the following assumptions:

- The time slice which is closer to the present time is more likely to predict current missing *QoS* accurately.
- Two users who are not really similar, may happen to have similar *QoS* experiences and will be mistaken as similar users. While, two users who are really similar, can hardly be mistaken as non-similar users.

In a specific time slice, the time interval is very small and the surrounding network environment usually keeps the

same. Hence, *QoS* values of different user-service pairs in the same time slice will have high relevance with each other. Obviously, based on historical *QoS* values, we can perform similar users(or services) selection with higher accuracy. By computing the user(or service) similarity at the level of time slice, we can finally find more confident similar users(or services), and further obtain accurate missing *QoS* prediction.

In this section, we will introduce our proposed approach in detail. Fig. 3 shows the procedure of our proposed approach. Each time slice of *QoS* matrix is normalized and transmitted into the system respectively as input. Firstly, we compute the average value of historical *QoS* through *Temporal Missing QoS Forecast*. In addition, *Similarity Computation Module* is responsible for computing the users(or services) similarity. Then, $TopN$ users(or services) with largest similarities will be selected by *Similar Neighbor Selection*, in order to perform the following *CF*-based missing *QoS* prediction with *QoS* matrix at T_1. Finally, we combine the above two values to get the final *QoS* prediction result.

If $R(u,s)$ is not an empty set, we can compute the average value of historical *QoS*, which can help to forecast the current *QoS*. The corresponding temporal forecasting value of the current user is calculated as follows,

$$\widetilde{r}_{tf}(u,s) = \frac{\sum_{r(u,s)\in R(u,s)} r(u,s)}{|R(u,s)|} \tag{9}$$

where $|R(u,s)|$ denotes the number of elements in $R(u,s)$. If $|R(u,s)| = 0$, we will set $\widetilde{r}_{tf}(u,s) = 0$.

4.1 Similarity Computation

In order to predict missing *QoS* values, we should firstly get the corresponding active users. As defined in *Definition* 3, we select users who have invoked the current service during the first time slice T_1.

It is worth to note that, similar users are not only the users who has almost the same *QoS*. Two users with similar *QoS* pattern should also be considered as similar users. Hence, we use *PCC* to compute the similarity. However, there are two cases that we should pay attention to in the *PCC* based similarity computation.

4.1.1 QoS Normalization

Suppose there are two users u_1 and u_2, their coinvoked services' *QoS* (here is response time) sets are as follows,

$$R(u_1, s|s \in S(u_1) \cap S(u_2)) = \{20, 50, 20, 50, 20, 50, 20, 50\}$$
$$R(u_2, s|s \in S(u_1) \cap S(u_2)) = \{50, 20, 50, 20, 50, 20, 50, 20\}$$

We can calculate the similarity between u_1 and u_2, according to Formula. 1.

$$Sim(u_1, u_2) = -1.0$$

Assume that u_1 and u_2 both invoke another service, which has a very large response time (i.e., 200). Then, their updated *QoS* sets are as follows,

$$R(u_1, s|s \in S(u_1) \cap S(u_2)) = \{20, 50, 20, 50, 20, 50, 20, 50, 200\}$$
$$R(u_2, s|s \in S(u_1) \cap S(u_2)) = \{50, 20, 50, 20, 50, 20, 50, 20, 200\}$$

Then, we can calculate the current similarity between u_1 and u_2 again, according to Formula. 1.

$$Sim(u_1, u_2) = 0.86$$

As we can see, u_1 and u_2 are quite irrelevant. But, just when they have coinvoked a service with larger response time, u_1 and u_2 are mistaken for similar users after *PCC*-based similarity computation. Hence, in order to resolve this issue, we perform a *QoS* normalization on every service.

$$r'(u,s) = \begin{cases} \dfrac{r(u,s) - r_{min}(s)}{r_{max}(s) - r_{min}(s)}, & r(u,s) \text{ is positive} \quad (10) \\ \dfrac{r_{max}(s) - r(u,s)}{r_{max}(s) - r_{min}(s)}, & r(u,s) \text{ is negative} \quad (10') \end{cases}$$

where $r_{min}(s)$ represents the minimize *QoS* value of s invoked by users in $U(s)$, and $r_{max}(s)$ represents the maximize *QoS* value of s invoked by users in $U(s)$. A positive *QoS* means that the larger the value is, the better the *QoS* (e.g., availability), while a negative *QoS* means that the larger the value is, the worse the *QoS* (e.g., response time). Apparently $r'(u,s)$ is the normalized $r(u,s)$.

After the normalization, a better *QoS* will be set with a larger value. In real applications, in order to satisfy users' requirements, services usually have relatively better *QoS*, which will be set with a larger value after normalization. However, a network failure or regression may happen by chance and further bring worse *QoS*, which will be set with a smaller value after normalization. In this way, we can decrease the affect on similarity computation coming from network failure or regression. In most cases, $0 \leq r'(u,s) \leq 1$. However, there may exist the situation that new *QoS* value is larger than the current maximum value. Then, $r'(u,s)$ will be larger than 1, which is also acceptable in our algorithm.

4.1.2 Weighted PCC

Suppose there are two users u_1 and u_2, their invoked service sets are $S(u_1)$ and $S(u_2)$ respectively. In most cases, *PCC* can provide accurate similarity computation. However, it may overestimate the similarities of users who are actually not similar but happen to have similar *QoS* experiences on a few coinvoked services. To address this problem, we employ a weight to reduce the influence of a small number of coinvoked items.

$$w(u,v) = \frac{2 \cdot |S(u) \cap S(v)|}{|S(u)| + |S(v)|} \tag{11}$$

where $|S(u) \cap S(v)|$ is the number of service items that are employed by both user u and user v, and $|S(u)|$ and $|S(v)|$ are the number of services invoked by user u and user v, respectively. Therefore, the value of $w(u,v)$ is in the interval of [0,1]. When the coinvoked service number $|S(u) \cap S(v)|$ is small, the significance weight $w(u,v)$ will

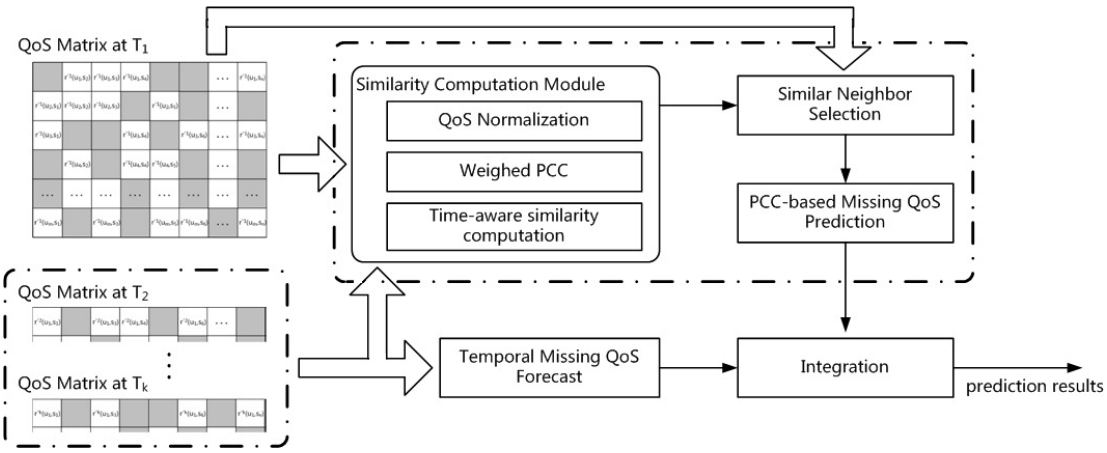


Fig. 3: Collaborative Filtering based Missing *QoS* Prediction.

decrease the similarity estimation between the users u and v, vice versa.

Just like the user-based methods, the weight for the similarity computation between different service items is defined as,

$$w(s,f) = \frac{2 \cdot |U(s) \cap U(f)|}{|U(s)| + |U(f)|} \tag{12}$$

where $|U(s) \cap U(f)|$ is the number of user items that employed both service s and service f, and $|U(s)|$ and $|U(f)|$ are the number of users invoked service s and service f, respectively. When the user number $|U(s) \cap U(f)|$ is small, the significance weight $w(s,f)$ will decrease the similarity estimation between the service s and f. The value of $w(s,f)$ is in the interval of [0,1].

4.1.3 Time-aware Similarity Computation

In order to reduce the affection of temporal factors, we should compute the users' or services' similarity for each time slice.

User based collaborative filtering adopts *PCC* (i.e., *UPCC*) [18] to compute the similarity between user u and user v,

$$Sim^{\theta}_{upcc}(u,v) = \frac{\sum_{s \in \bar{S}^{\theta}} (r'^{\theta}(u,s) - \overline{r'^{\theta}(u)}) \cdot (r'^{\theta}(v,s) - \overline{r'^{\theta}(v)})}{\sqrt{\sum_{s \in \bar{S}^{\theta}} (r'^{\theta}(u,s) - \overline{r'^{\theta}(u)})^2} \cdot \sqrt{\sum_{s \in \bar{S}^{\theta}} (r'^{\theta}(v,s) - \overline{r'^{\theta}(v)})^2}} \tag{13}$$

where $Sim^{\theta}(u,v)$ is the similarity of user u and user v according to the *QoS* values at time slice T_θ, which is in the interval of [-1, 1], $\widetilde{S}^{\theta} = S^{\theta}(u) \cap S^{\theta}(v)$ is the set of services which have been invoked by both user u and user v, $\overline{r'^{\theta}(u)}$ and $\overline{r'^{\theta}(v)}$ represent the average *QoS* values of different services which have been invoked by user u and user v respectively. When two users have null service intersection (i.e., $\widetilde{S}^{\theta} = NULL$), the corresponding similarity will be ignored in the following procedure.

We can get the user similarity with the weight computed according to Equation. 11.

$$Sim^{\theta}(u,v) = w^{\theta}(u,v) \cdot Sim^{\theta}_{upcc}(u,v) \tag{14}$$

As mentioned previously, users may happen to perceive similar *QoS* values on a few services. But they are not really similar. By integrating the computed similarities from k time slices, we can reduce the affect coming from the above mentioned issue, and select the real similar users.

We assign a specific weight to each computed $Sim^{\theta}(u,v)$, and compute the weighted sum as follows,

$$Sim(u,v) = \frac{\sum_{\theta=1}^{k} p^{\theta}(u,v) * Sim^{\theta}(u,v)}{\sum_{\theta=1}^{k} p^{\theta}(u,v)} \tag{15}$$

For the specific value of $p^{\theta}(u,v)$, we have the following two assumptions,

- The time slice which is closer to the present time is more likely to predict current missing *QoS* accurately.
- If similar users (or services) at one time slice have higher similarity, then the predicted missing *QoS* based on these users (or services) will be more accurate.

Based on the above two assumptions, we get the weight as follows,

$$p^{\theta}(u,v) = \frac{N(\widetilde{S}^{\theta})}{log_n(k+1)} \tag{16}$$

where $N(\widetilde{S}^{\theta})$ is the number of service set $\widetilde{S}^{\theta} = S^{\theta}(u) \cap S^{\theta}(v)$, $\frac{1}{log_n(k+1)}$ indicates that larger k will result in smaller weight.

In the same way, service based collaborative filtering adopts *PCC* (i.e., *IPCC*) [19] to compute the similarity between service s and service f,

$$Sim^{\theta}_{ipcc}(s,f) = \frac{\sum_{u \in \bar{U}^{\theta}} (r'^{\theta}(s,u) - \overline{r'^{\theta}(s)}) \cdot (r'^{\theta}(f,u) - \overline{r'^{\theta}(f)})}{\sqrt{\sum_{u \in \bar{U}^{\theta}} (r'^{\theta}(s,u) - \overline{r'^{\theta}(s)})^2} \cdot \sqrt{\sum_{u \in \bar{U}^{\theta}} (r'^{\theta}(f,u) - \overline{r'^{\theta}(f)})^2}} \tag{17}$$

where $Sim^{\theta}(s,f)$ is the similarity of service s and f according to the *QoS* values at time slice T_θ, which is in the interval of [-1, 1], $\widetilde{U}^{\theta} = U^{\theta}(s) \cap U^{\theta}(f)$ is the set of users who have invoked both service s and service f, $\overline{r'^{\theta}(s)}$ and $\overline{r'^{\theta}(f)}$ represent the average *QoS* values of different users who have invoked service s and f respectively. When two

services have null user intersection (i.e., $\widetilde{U}^{\theta} = NULL$), the corresponding similarity will also be ignored in the following procedure.

Similarly, we can get the service similarity with the weight computed according to Equation. 12.

$$Sim^{\theta}(s,f) = w^{\theta}(s,f) \cdot Sim^{\theta}_{ipcc}(s,f) \tag{18}$$

We assign a specific weight to each computed $Sim^{\theta}(s,f)$, and compute the weighted sum as follows,

$$Sim(s,f) = \frac{\sum_{\theta=1}^{k} p^{\theta}(s,f) * Sim^{\theta}(s,f)}{\sum_{\theta=1}^{k} p^{\theta}(s,f)} \tag{19}$$

$$p^{\theta}(s,f) = \frac{N(\widetilde{U}^{\theta})}{log_n(k+1)} \tag{20}$$

where $N(\widetilde{U}^{\theta})$ is the number of user set $\widetilde{U}^{\theta} = U^{\theta}(s) \cap U^{\theta}(f)$.

4.2 Similar Neighbor Selection

Selecting the users (or services) right similar to the active user is necessary for accurate missing *QoS* prediction. In conventional *CF*, the $TopN$ similar neighbor selection algorithm is often employed. It selects N users that are most similar to the active user as his/her neighbors. We adopt the $TopN$ similar neighbor selection algorithm in our similar users (or services) selection.

In practice, some users (or services) may have a few similar users (or services) or no similar users (or services) due to data sparsity. However, traditional $TopN$ similar neighbor selection algorithm ignore this issue and still select the top N most similar ones. As the selected users (or services) are not actually similar to the current users (or services), the final predicted *QoS* has lower accuracy. Hence, we propose a two-layer filtering mechanism. The first threshold ψ^{u}_{sim1} is used to filter similar users based on the computed similarity according to Equation 15, while the second threshold ψ^{u}_{sim2} is used to filter similar users from each time slice based on the computed similarity according to Equation 14. Firstly, removing users (or services) from similar users (or services) set is better if the similarity is no more than a given threshold ψ^{u}_{sim1}.

$$SU_{thres}(u) = \{v | Sim(u,v) > \psi^{u}_{sim1}, u \neq v\} \tag{21}$$

where $SU_{thres}(u)$ represents the selected candidated similar users.

Meanwhile,

$$SS_{thres}(s) = \{f | Sim(s,f) > \psi^{s}_{sim1}, s \neq f\} \tag{22}$$

where $SS_{thres}(s)$ represents the selected candidate similar services.

Let $TopN(u)$ denote the resulting subset of similar neighbors $SU_{thres}(u)$ for the current user u. Then, we construct similar user set $TopN(u)$ based on $SU_{thres}(u)$ as follows:

- Step 1: Among $SU_{thres}(u)$, we can obtain the active user set $AU^1(u)$ of user u in time slice T_1. Calculate the similarity between user u and each user in $AU^1(u)$, according to Equation 15. Then, we can get $SU^1_{thres}(u) = \{v | Sim(u,v) > \psi^{u}_{sim2}, u_0 \neq v\}$ from $AU^1(u)$. Finally, search $SU^1_{thres}(u)$ for $TopN$ most similar users. If the size of $SU_{TopN}(u)$ is smaller than $TopN$, proceed to Step 2, else return $SU_{TopN}(u)$.
- Step 2: Go to the next time slice, and take the current time slice as the first one, then the following time slices number will also be changed in turn. Based on the new *QoS* matrix, we perform the above mentioned time-aware similarity computation process and then get a new $SU_{thres}(u)$. Among the new $SU_{thres}(u)$, we can obtain the active user set $AU^1(u)$ of user u in the current new time slice T_1. Calculate the similarity between user u and each user in $AU^1(u)$, according to Equation 15. Then, we can get new $SU^1_{thres}(u) = \{v | Sim(u,v) > \psi^{u}_{sim2}, u \neq v\}$ from $AU^1(u)$. Finally, search $SU^1_{thres}(u)$ for $TopN$ most similar users. If the size of $SU_{TopN}(u)$ is smaller than $TopN$ and smaller than $\frac{5}{3} \times |SU^0_{TopN}(u)|$ ($|SU^0_{TopN}(u)|$ is the size of $SU_{TopN}(u)$ in Step 1) meanwhile, proceed to Step 3, else return $SU_{TopN}(u)$.
- Step 3: For each other time slice, repeat Step 2 until $TopN$ most similar users are found, or all the k time slices are searched. The obtained $SU(u)$ is the final similar neighbor user set, then return $SU(u)$.

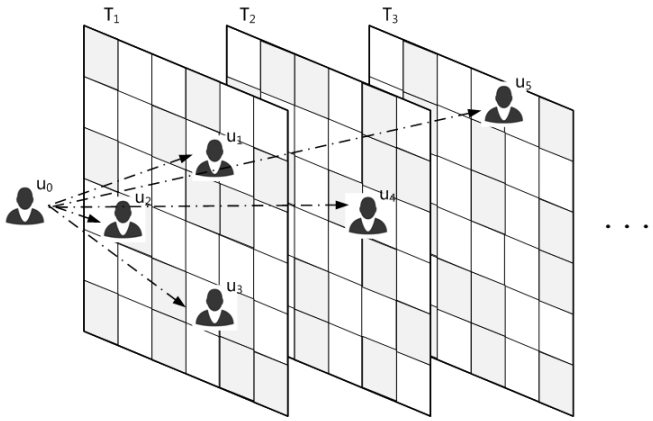


Fig. 4: Similar Neighbor Selection.

Fig. 4 is an example illustrating how we select similar neighbors for user u_0. We set $TopN = 5$. Suppose u_1, u_2 and u_3 are the only three similar neighbors of u_0 in T_1 (i.e., $SU_{TopN}(u_0) = \{u_1, u_2, u_3\}$), we continue to search T_2 for similar neighbors, as the number of selected similar neighbors is smaller than 5. Suppose u_4 are the most similar neighbors of u_0, then $SU_{TopN}(u_0) = \{u_1, u_2, u_3, u_4\}$. We continue to search T_3 for similar neighbors, as the number of selected similar neighbors is still smaller than 5. In the third round, suppose u_5 are the most similar neighbors of u_0, then $SU_{TopN}(u_0) = \{u_1, u_2, u_3, u_4, u_4\}$, the similar neighbor selection will be stopped, as we have selected $TopN$ similar neighbors successfully.

In some cases, there may be not enough similar users. Hence, the size of $SU_{TopN}(u_0)$ may be smaller than $TopN$. In this way, we can further get the similar neighbor services

$SS_{TopN}(s_0)$.

4.3 Missing QoS Prediction

With the *QoS* values from similar neighbors, we can perform *CF*-based *QoS* prediction.

Based on $SU_{TopN}(u)$, *UPCC* predicts the missing *QoS* values by the following equation,

$$\widetilde{r}_{upcc}(u,s) = \overline{r^1(u)} + \frac{\sum_{v \in SU(u)}(Sim(u,v) \cdot (r^1(v,s) - \overline{r^1(v)}))}{\sum_{v \in SU(u)}(Sim(u,v))} \tag{23}$$

where $\widetilde{r}_{upcc}(u,s)$ means the predicted *QoS* values based on *UPCC*, $\overline{r^1(u)}$ denotes the average *QoS* values of user u at time slice T_1, $\overline{r^1(v)}$ denotes the average *QoS* values of user v at time slice T_1, and $r^1(v,s)$ denotes the *QoS* value of user v at time slice T_1.

Meanwhile, based on $SS_{TopN}(s)$, *IPCC* predicts the missing *QoS* values by the following equation,

$$\widetilde{r}_{ipcc}(u,s) = \overline{r^1(s)} + \frac{\sum_{f \in SS(s)}(Sim(s,f) \cdot (r^1(f,u) - \overline{r^1(f)}))}{\sum_{f \in SS(s)}(Sim(s,f))} \tag{24}$$

where $\widetilde{r}_{ipcc}(u,s)$ means the predicted *QoS* values based on *IPCC*, $\overline{r^1(s)}$ denotes the average *QoS* values of service s at time slice T_1, $\overline{r^1(f)}$ denotes the average *QoS* values of service f at time slice T_1 and $r^1(u,f)$ denotes the *QoS* value of service f at time slice T_1.

By combining *UPCC*-based prediction and *IPCC*-based prediction, we can get the final *CF*-based predicted *QoS* value as follows.

$$\widetilde{r}_{cf}(u,s) = \lambda \cdot \widetilde{r}_{upcc}(u,s) + (1-\lambda) \cdot \widetilde{r}_{ipcc}(u,s) \tag{25}$$

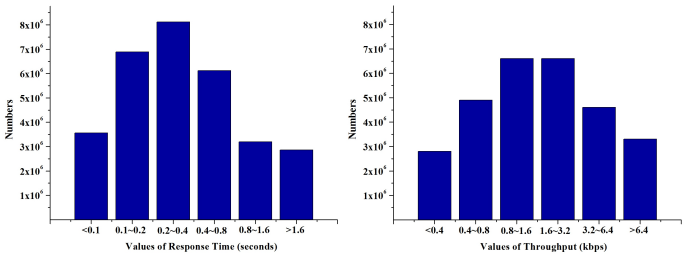
where $0 \leq \lambda \leq 1$, it decides how much the *CF*-based predicted *QoS* value relies on $\widetilde{r}_{upcc}(u,s)$ or $\widetilde{r}_{ipcc}(u,s)$. If $\lambda = 0$, only $\widetilde{r}_{ipcc}(u,s)$ is involved in *QoS* prediction. If $\lambda = 1$, only $\widetilde{r}_{upcc}(u,s)$ is involved in *QoS* prediction. In other cases, $\widetilde{r}_{upcc}(u,s)$ and $\widetilde{r}_{ipcc}(u,s)$ are combined to predict the missing *QoS*.

To fully utilize the prediction values from temporal forecasting and collaborative filtering, we combine them and get the final prediction result:

$$\widetilde{r}(u,s) = \begin{cases} \mu \cdot \widetilde{r}_{cf}(u,s) + (1-\mu) \cdot \widetilde{r}_{tf}(u,s), \ \widetilde{r}_{cf}(u,s) \neq 0 \ and \ \widetilde{r}_{tf}(u,s) \neq 0; \\ \widetilde{r}_{cf}(u,s), \quad \widetilde{r}_{tf}(u,s) = 0 \ and \ \widetilde{r}_{cf}(u,s) \neq 0; \\ \widetilde{r}_{tf}(u,s), \quad \widetilde{r}_{tf}(u,s) = 0 \ and \ \widetilde{r}_{cf}(u,s) \neq 0; \\ 0, \quad \widetilde{r}_{tf}(u,s) = 0 \ and \ \widetilde{r}_{cf}(u,s) = 0; \end{cases} \tag{26}$$

where $0 \leq \mu \leq 1$, it decides how much the final predicted *QoS* value relies on $\widetilde{r}_{tf}(u,s)$ or $\widetilde{r}_{cf}(u,s)$. When the network environment is much more dynamic, μ should be set larger than 0.5, vice versa.

When the predicted value of the current user for the current service has been obtained, the recommendation system can recommend services according to users' functional and non-functional constraints.



(a) Response Time (b) Throughput

Fig. 5: QoS value Distributions.

5 Experiments

In this section, we conduct a set of experiments, in order to address the following issues: (1)The prediction accuracy compared with existing prediction approaches. (2)Impact of parameter settings.

Experiments were performed on Lenovo desktop computer with Intel Core i7 3.2GHz CPU and 8GB RAM. In this section, we first describe the experimental setup. Then, experiments are performed to evaluate the prediction accuracy compared with other collaborative filtering approaches. Finally, we discuss the experimental results in detail.

5.1 Experimental Setup

Before the experiments, we should firstly get enough dataset of users and services with their corresponding *QoS* values. *WS-DREAM* [10] maintains a set of *QoS* datasets which are collected from real world web services. In our experiments, we adopted the time-aware *QoS* dataset of *WS-DREAM*. This dataset describes *QoS* records (*response time* and *throughput*) of service invocations on $4,500$ web services from 142 users over 64 consecutive time slices (at 15-minute interval).

TABLE 2: Temporal QoS Data

Statistic	Response Time	Throughput
Scale	0-20s	0-1000kbps
Mean	3.165s	9.609kbps
Number of Users	142	142
Number of Services	4532	4532
Number of Time Intervals	64	64
Number of Records	30287611	30287611

The statistics of web service *QoS* dataset are summarized in Table. 2. Response-time and throughput are within the valid range of 0-20 seconds and 0-1000 kbps respectively. The means of response-time and throughput are 3.165 seconds and 9.609 kbps respectively. The distributions of the response-time and throughput values of the user-service time tensors are shown in Fig. 5a and Fig. 5b respectively. Most of the response-time values are between 0.1-0.8 seconds and most of the throughput values are between 0.8-3.2 kbps.

In addition, we find that in *WS-DREAM* dataset, there exist some response-time values that are 20.0 or 19.999. It's

very possible that these *QoS* values are specially set for data validation. Hence, in order to make the further prediction accurately, we first take a preprocessing to remove all these *QoS* values.

The parameter setting is shown in Table. 3.

TABLE 3: Parameter Setting

Parameters	Values
$dataset\ density$	{0.05, 0.1, 0.15, 0.2, 0.25}
T	{16, 32, 48, 64}
k	{4, 8, 16}
λ	{0.1, 0.3, 0.5, 0.7, 0.9}
$TopN$	{10, 20, 30, 40, 50}
μ	0.62

5.2 Evaluation Metric

The goal of prediction methods is to minimize the deviation between the predicted value and the observed value. In addition, the number of predicted *QoS* values with smaller deviation is also an important factor for evaluating missing *QoS* prediction methods. Hence, We evaluate the performance of different prediction methods on three measures: (1) average mean absolute error (*MAE*) defined as Equation 27; (2) average root-mean-square error (*RMSE*) defined as Equation 28; (3) average P_{20} defined as Equation 29. A lower *MAE* and *RMSE* indicates a better performance of the method, while a higher P_{20} indicates a better performance of the method.

$$MAE = \frac{\sum |r(u,s) - \tilde{r}|}{N} \tag{27}$$

$$RMSE = \sqrt{\frac{\sum (r(u,s) - \tilde{r})^2}{N}} \tag{28}$$

where $r(u,s)$ denotes the observed *QoS* value of service s invoked by user u, $\tilde{r}$ is the predicted *QoS* value, N is the number of predicted values. *MAE* and *RMSE* are also frequently used to measure the difference between values predicted by a model and observed values.

$$P_{20} = \frac{N_{20}}{N} \tag{29}$$

where N is the number of predicted values, and N_{20} is the number of predicted values that are within 20% deviation compared to the observed *QoS* values.

5.3 Evaluation on Prediction Accuracy

To evaluate the performance of our proposed approach, we compare our approach with the following well known *QoS* prediction methods.

- *UPCC (user-based collaborative filtering using PCC).* *UPCC* employs similar users to predict the missing *QoS* values.
- *IPCC (item-based collaborative filtering using PCC).* *IPCC* employs similar services (items) to predict the missing *QoS* values.
- *WSRec* [37]. *WSRec* is a hybrid collaborative algorithm that combines both *UPCC* and *IPCC* approaches to predict the missing *QoS* values.
- *K-SLOPE* [36]. *K-SLOPE* uses K-means to exclude the less similar users. Slope One algorithm is also adopted to predict the missing *QoS* over time.
- *TMF* [31]. *TMF* is a time-aware matrix factorization (*TMF*) model, which uses an adaptive matrix factorization to predict the missing *QoS* values. A temporal smoothing method is then applied to the predicted result to perform the time-varying *QoS* prediction.
- *NTF* [29]. *NTF* (Non-negative Tensor Factorization) is a generalized tensor factorization model, which replaces the user-service matrix in matrix factorization with the user-service-time interaction relations by considering the difference of *QoS* value at difference time.
- *RTF* [32]. *RTF* (Recurrent Tensor Factorization) is a Deep Learning based time-aware service prediction framework, which combines Tensor Factorization (*TF*) with Deep Learning (*DL*) for time-aware service recommendation.
- *TF-KMP* [30]. *TF* takes the average value of temporal historical values as the forecasted future *QoS*. *KMP* is a collaborative filtering *QoS* prediction method based on *K*-means clustering. *TF-KMP* combines *TF* and *KMP* to predict the missing *QoS* values.

In practice, service invocations are not always consecutive. It is inevitable that there will be missing *QoS* records. Hence, the user-service-time *QoS* matrix is sparse. Indeed, the time-aware *QoS* dataset of *WS-DREAM* also contains missing *QoS* records. Statistically, the initial densities of *WS-DREAM* are shown in Fig. 6.

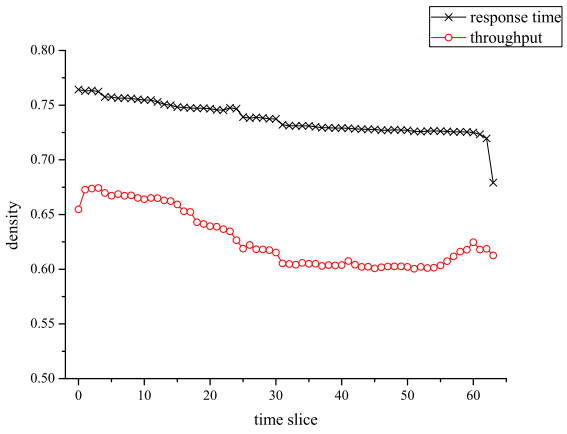


Fig. 6: Initial Densities of *WS-DREAM*.

As we can see in Fig. 6, the maximize initial density of *response time* dataset is 76%, and the minimize initial density of *response time* dataset is 68%. While the maximize initial density of *throughput* dataset is 67%, and the minimize initial density of *throughput* dataset is 60%. Furthermore, in order to simulate various application cases, we will randomly remove specified number of *QoS* values from

WS-DREAM dataset to get a set of datasets with various densities.

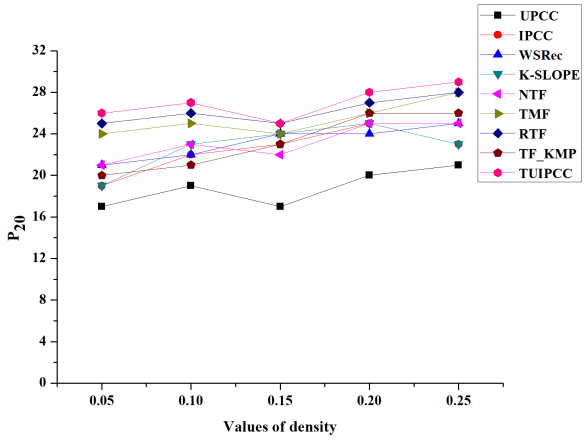
In our experiments, the density parameter of dataset is set to 0.05, 0.1, 0.15, 0.2 and 0.25 respectively. In order to simulate various application cases, we randomly remove specified number of *QoS* values from *WS-DREAM* dataset several times to get a set of datasets with various densities. Furthermore, a set of experiments can be performed based on these datasets to validate the generalizability. We randomly select 100 user-service pairs that *QoS* is missing to perform prediction. Each experiment case will be run 10 times and average *MAE* and *RMSE* will be computed. Then, experimental results are shown in Table. 4 and Fig. 7. For simplicity, we use *TUIPCC* to denote our proposed approach. The detailed investigations of parameter setting will be provided in Section 5.4.

As we can see in Table. 4, *CF*-based approaches all have satisfactory *MAE* and *RMSE*. To sum up, our proposed approach has better *MAE* and *RMSE* compared with *UPCC*, *IPCC*, *WSRec*, *K-SLOPE*, *NTF*, *TMF*, *RTF* and *TF-KMP*. *MAE* and *RMSE* are not obvious regularity as the increasing of matrix density, indicating that the prediction accuracy may be partially influenced by the application environment. However, from the tendency, despite some exception, all approaches perform better as *QoS* matrix density increases, because when *QoS* matrix density is larger, more similar users (or services) can be employed for prediction.

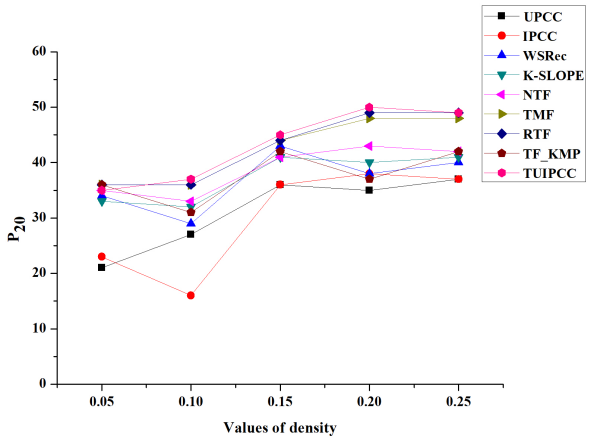
Experimental results show that the performance improvement of our proposed approach over the best baseline is often subtle. Indeed, missing *QoS* prediction is usually a difficult problem. On one hand, the accurate latent representations extraction is difficult. On the other hand, the network environment is dynamic. There exists many factors that can effect the final *QoS*. *QoS* of services is strongly linked to the service status and network environments, which are variable against time. Hence, as shown in Table. 4, although the performance improvement of our proposed approach is subtle, our proposed approach has better *MAE* and *RMSE* in different parameter setting, generally.

As we can see in Fig. 7, among all the seven approaches, our proposed *TUIPCC* has the largest P_{20}. That means *TUIPCC* has larger probability to predict missing *QoS* values with smaller deviation.

Overall, *UMean* does not take into account the network environment information. The values of *response time* and *throughputs* fluctuate greatly when the user invokes different services, because the user's location is fixed while different services are deployed at different places. Similar to *UMean*, *IMean* also has a large deviation. The better performance of our proposed *TUIPCC* compared to *UPCC*, *IPCC* and *WSRec* validates the usefulness of temporal historical *QoS* in missing *QoS* prediction. *TUIPCC* integrated similarity information from a set of time slices, which can find real similar users (or services) with higher probability. Therefore, we can achieve better performance than *TF-KMP*. In addition, *TUIPCC* considers more about the dynamic characteristic of network environment, and makes good use of temporal historical *QoS* data. Hence, we can achieve better performance than existing time-aware *QoS* prediction approaches.



(a) Response Time



(b) Throughput

Fig. 7: The number of predicted *QoS* values with smaller deviation.

5.4 Impact of Parameter Setting

5.4.1 *Impact of T and k*

In order to get accurate *QoS* prediction, the selection of appropriate value of T and k pair is very important. Larger T will bring more outdated *QoS* values, which may affect the *QoS* prediction accuracy. While, smaller T may filter out more valuable historical information. In addition, the value of k should also be selected carefully, according to the specific *QoS* property and application environment. For example, when the application scenario is more dynamic, the value of k should be smaller, vice versa.

The parameter settings are *density=20%* and $\lambda = 0.9$. The value of T various from 16 to 64 with a step value of 16, and the value of k will be set to 4, 8 and 16 respectively. Fig. 8 shows the experiment results.

As we can see in Fig. 8, in our experiment scenario, T and k should be set with a larger value, i.e., $T = 64$ and $k = 16$.

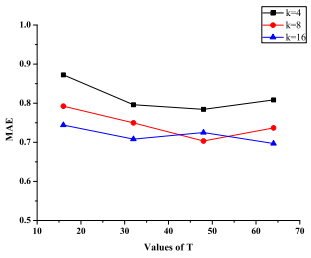
5.4.2 *Impact of λ*

Parameter λ decides how much our approach relies on *UPCC* or *IPCC*.

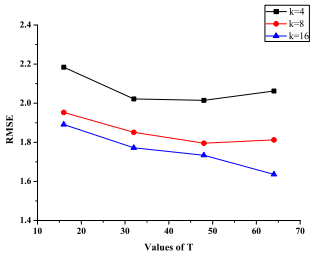
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2021.3103769, IEEE Transactions on Services Computing

TABLE 4: Demographic Prediction performance comparison.

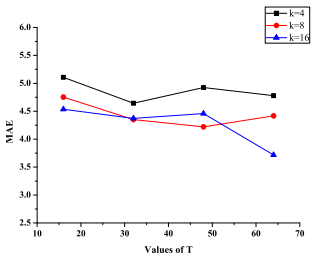
	Methods	Matrix Density = 5%		Matrix Density = 10%		Matrix Density = 15%		Matrix Density = 20%		Matrix Density = 25%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Response Time	UPCC	0.7971	1.9173	0.5463	1.3283	0.9670	2.4075	0.7583	1.7333	0.4876	0.9350
	IPCC	0.8928	2.1562	0.6324	1.3483	0.9978	2.3594	0.7913	1.7944	0.4890	0.9833
	WSRec	0.8075	1.9479	0.5491	1.3247	0.9613	2.4012	0.7625	1.7258	0.4001	0.7971
	K-SLOPE	0.8127	1.9363	0.6082	1.3394	0.8581	2.3516	0.7536	1.7115	0.4621	0.8625
	NTF	0.8016	1.8768	0.5811	1.4375	0.8449	2.3543	0.7458	1.8168	0.4616	0.8304
	TMF	0.7801	1.7698	0.5802	1.4079	0.8315	2.1847	0.7132	1.6593	0.3976	0.7863
	RTF	0.7896	1.8613	0.5772	1.2218	0.8253	2.0935	0.6917	1.6419	0.3801	0.7395
	TF-KMP	0.7923	1.8734	0.5783	1.4235	0.8584	2.3671	0.7404	1.8292	0.4609	0.8448
	TUIPCC	**0.7814**	**1.7761**	**0.5767**	**1.2076**	**0.8196**	**2.0595**	**0.6970**	**1.6358**	**0.3725**	**0.7284**
Throughput	UPCC	4.6554	7.8959	5.4148	18.6715	6.6406	22.0341	5.7306	20.2794	4.1267	10.2451
	IPCC	7.1970	13.9509	6.8750	19.0509	7.9934	18.1130	7.0910	20.6741	7.6103	16.7427
	WSRec	4.7834	7.8885	5.9324	18.3218	7.0116	21.3235	6.1093	19.9576	4.4261	10.3490
	K-SLOPE	4.8273	8.3239	4.9438	16.1546	6.5173	23.5746	4.7596	16.1765	4.5381	11.5274
	NTF	4.8817	10.0173	4.7668	16.3087	5.2548	22.9082	3.9223	14.2805	3.7669	8.1501
	TMF	4.8526	9.2550	4.2371	14.5402	5.3402	23.5708	3.8105	13.5441	3.2479	7.5624
	RTF	4.8419	9.1748	3.8564	13.6506	5.2337	22.6301	3.7362	12.4158	3.0362	7.0139
	TF-KMP	4.8642	9.3827	4.5507	15.2182	5.2929	23.8529	4.0706	14.8778	3.9715	9.6375
	TUIPCC	**4.8399**	**10.1350**	**3.7573**	**13.2691**	**5.2349**	**23.2172**	**3.7177**	**12.2145**	**3.0223**	**6.9653**



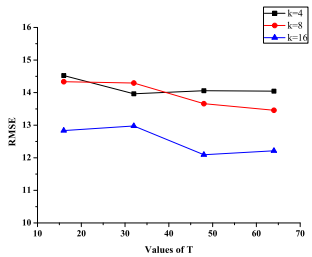
(a) Response Time



(b) Response Time

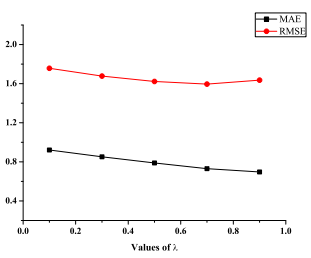


(c) Throughput

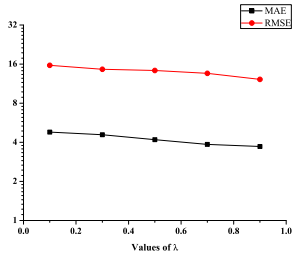


(d) Throughput

Fig. 8: Prediction Accuracy Comparison Under Different T and k.



(a) Response Time



(b) Throughput

Fig. 9: Prediction Accuracy Comparison Under Different λ.

In this section, we will do experiments to evaluate the impact of different λ. The parameter settings are *density=20%*, *T=64* and *k=16*. The value of λ various from 0.1 to 0.9 with a step value of 0.2. Fig. 9 shows the impacts of parameter λ on the prediction accuracy.

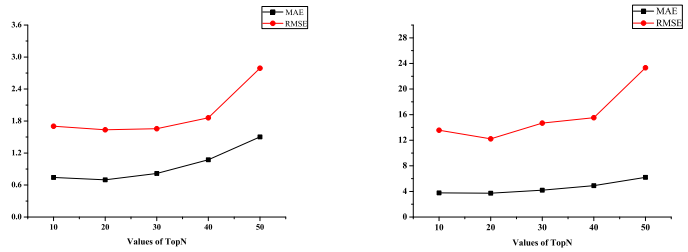
We observed that, by assigning λ with an appropriate value, we can achieve better prediction accuracy. As we can see in Fig. 9, in our experiment case, our approach achieves better performance when $\lambda = 0.9$. Obviously, the value of λ should be selected according to the specific application scene.

5.4.3 *Impact of* $TopN$

Parameter $TopN$ decides how many similar users (or services) will be employed to perform missing *QoS* prediction.

In our experiments, the parameter settings are *density=20%*, *T=64*, *k=16* and $\lambda = 0.9$. We set $TopN$ with 10, 20, 30 40 and 50 respectively. In order to evaluate the impact of $TopN$ successfully, we cancel the usage of ψ^s_{sim1} and ψ^s_{sim2}. That means, we will not perform the filtering process in Sec. 4.2. Then, we can get enough $TopN$ similar neighbors. Fig. 10 shows the prediction results.

As we can see in Fig. 10, when $TopN$ exceeds 30, the prediction accuracy declines quickly. We believe that, with appropriately larger $TopN$, more users (or services) will be employed, and accordingly the prediction accuracy will be higher. However, when $TopN$ is larger than a specific value, users (or services) with low similarity will be inevitably



(a) Response Time (b) Throughput

Fig. 10: Prediction Accuracy Comparison Under Different $TopN$.

employed to perform missing *QoS* prediction, which will largely decrease the prediction accuracy. Hence, $TopN$ is not the larger the better, and we should find the optimal $TopN$ according to the specific application scene.

5.5 Discussions

In the experiment section, we verify the prediction performance of our proposed approach. Compared with existing prediction methods, our proposed approach has better performance. On one hand, the proposed approach filters out the historical *QoS* values with good timeliness, which can represent the current network environment accurately. On the other hand, the proposed approach takes into account the valuable information from temporal historical *QoS* values. Then, through time-aware collaborative filtering mechanism, we can achieve accurate missing *QoS* prediction.

In order to achieve better performance, some parameters should be considered carefully. Experiment results show that T, K, λ, and $TopN$ will all affect the final prediction accuracy. In order to remove the outdated *QoS* values, T should be set with a relatively small value. In addition, K decides the size of time slices, λ determines how much the predicted *QoS* relies on user-based prediction or service-based prediction, and $TopN$ is responsible for limiting similar neighbor searching to a small subset of users or services. As we know, different application scenarios have different data correlation characteristics. Therefore, K, λ and $TopN$ should be set properly to match the specific application scenario.

6 Conclusions

Missing *QoS* prediction is an important issue in *QoS*-based service discovery and selection. In this paper, we proposed an improved collaborative filter based missing *QoS* prediction approach. Aiming at present the current application scenario accurately, we firstly proposed a time aware *QoS* model. It filtered out the historical *QoS* values with good timeliness, and then divided the obtained *QoS* values into several time slices. Subsequently, by integrating the similarity computation results at different time slices, we can minimize the affect coming from dynamic application environment and select the real similar users (or services). Finally, through hybrid collaborative filtering, we succeeded to predict missing *QoS* values. Experiments conducted on a real web service dataset indicated that our proposed approach outperforms previous *CF*-based prediction methods.

In the future, we will try to propose a method to select appropriate k automatically. In addition, some *QoS* attributes may have close relationship with each other, i.e., they may increase or decrease synchronously. Hence, we will further study the *QoS* attributes relationship and employ them to perform missing *QoS* prediction comprehensively.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61802389, 61972025), the Fundamental Research Funds for the Central Universities of China (2019RC008), the National Key Research and Development Program of China under Grant 2020YFB2103802 and Grant 2020YFB1005604.

References

[1] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 3, pp. 281–308, 2004.

[2] Y. Liu, A. H. Ngu, and L. Zeng, "Qos computation and policing in dynamic web service selection," in *Proceedings of 13th IEEE International Conference on World Wide Web*. IEEE, 2004, pp. 66–73.

[3] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of 18th IEEE International Conference on World Wide Web*. IEEE, 2009, pp. 881–890.

[4] J. E. Hadad, M. Manouvrier, and M. Rukoz, "Tqos: Transactional and qos-aware selection algorithm for automatic web service composition," *IEEE Transactions on Services Computing*, vol. 3, no. 1, pp. 73–85, 2010.

[5] K. Lee, J. Jeon, W. Lee, S. Jeong, and S. Park, "Qos for web services: Requirements and possible approaches," *W3C Working Group Note*, vol. 25, no. November, pp. 1–9, 2003.

[6] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "A privacy-preserving qos prediction framework for web service recommendation," in *Proceedings of 22nd IEEE International Conference on Web Services*. IEEE, 2015, pp. 241–248.

[7] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service qos prediction with location-based regularization," in *Proceedings of 19th IEEE International Conference on Web Services*. IEEE, 2012, pp. 464–471.

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

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

[10] Y. Zhang, Z. Zheng, and M. R. Lyu, "Wspred: A time-aware personalized qos prediction framework for web services," in *Proceedings of IEEE Symposium on Software Reliability Engineering*, 2011, pp. 210–219.

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

[12] Y. Hu, Q. Peng, X. Hu, and R. Yang, "Web service recommendation based on time series forecasting and collaborative filtering," in *Proceedings of IEEE International Conference on Web Services*. IEEE, 2015, pp. 233–240.

[13] R. Xiong, J. Wang, Z. Li, B. Li, and P. C. Hung, "Personalized lstm based matrix factorization for online qos prediction," in *2018 IEEE International Conference on Web Services (ICWS)*. IEEE, 2018, pp. 34–41.

[14] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, no. 1, pp. 1–24, 2010.

[15] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[16] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of International Conference on Uncertainty in Artificial Intelligence*. ACM, 1998, pp. 43–52.

[17] W. Lo, J. Yin, Y. Li, and Z. Wu, "Efficient web service qos prediction using local neighborhood matrix factorization," *Engineering Applications of Artificial Intelligence*, vol. 38, pp. 14–23, 2015.

[18] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proceedings of ACM Sigir Conference on Information Retrieval*. ACM, 2005, pp. 114–121.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of IEEE International Conference on World Wide Web*. IEEE, 2001, pp. 285–295.

[20] J. Wang, A. P. Vries, and M. J. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of ACM Sigir Conference on Information Retrieval*. ACM, 2006, pp. 501–508.

[21] "Pearson correlation coefficient," https://en.wikipedia.org/wiki/Pearson_correlation_coefficient, 2018.

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

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

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

[25] Y. Ma, S. Wang, P. C. K. Hung, C. Hsu, Q. Sun, and F. Yang, "A highly accurate prediction algorithm for unknown web service qos values," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 511–523, 2016.

[26] B. Cavallo, M. D. Penta, and G. Canfora, "An empirical comparison of methods to support qos-aware service selection," in *Proceedings of International Workshop on Principles of Engineering Service-Oriented Systems*, 2010, pp. 64–70.

[27] S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang, "Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and arima model," *Decision Support Systems*, vol. 107, pp. 103–115, 2018.

[28] X. Wu, Y. Fan, J. Zhang, H. Lin, and J. Zhang, "Qf-rnn: Qi-matrix factorization based rnn for time-aware service recommendation," in *2019 IEEE International Conference on Services Computing (SCC)*. IEEE, 2019, pp. 202–209.

[29] W. Zhang, H. Sun, X. Liu, and X. Guo, "Temporal qos-aware web service recommendation via non-negative tensor factorization," in *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 585–596.

[30] C. Wu, W. Qiu, X. Wang, Z. Zheng, and X. Yang, "Time-aware and sparsity-tolerant qos prediction based on collaborative filtering," in *Proceedings of IEEE International Conference on Web Services*. IEEE, 2016, pp. 637–640.

[31] S. Li, J. Wen, F. Luo, and G. Ranzi, "Time-aware qos prediction for cloud service recommendation based on matrix factorization," *IEEE Access*, vol. 6, pp. 77 716–77 724, 2018.

[32] Y. Zhang, C. Yin, Z. Lu, D. Yan, M. Qiu, and Q. Tang, "Recurrent tensor factorization for time-aware service recommendation," *Applied Soft Computing*, vol. 85, p. 105762, 2019.

[33] S. Chen, Y. Fan, W. Tan, J. Zhang, B. Bai, and Z. Gao, "Service recommendation based on separated time-aware collaborative poisson factorization." *J. Web Eng.*, vol. 16, no. 7&8, pp. 595–618, 2017.

[34] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, "Time-aware distributed service recommendation with privacy-preservation," *Information Sciences*, vol. 480, pp. 354–364, 2019.

[35] G. Tian, J. Wang, K. He, C. Sun, and Y. Tian, "Integrating implicit feedbacks for time-aware web service recommendations," *Information systems frontiers*, vol. 19, no. 1, pp. 75–89, 2017.

[36] M. Fayala and H. Mezni, "Web service recommendation based on time-aware users clustering and multi-valued qos prediction," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 9, p. e5603, 2020.

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