2018 IEEE International Conference on Services Computing

A Two-phase Method of QoS Prediction for Situated Service Recommendation

Jiapeng Dai, Donghui Lin and Toru Ishida
Department of Social Informatics
Kyoto University
Yoshida-Honmachi, Sakyo-ku, 606-8501, Kyoto, Japan
Email: dai-jiapeng@ai.soc.i.kyoto-u.ac.jp, {lindh, ishida}@i.kyoto-u.ac.jp

***Abstract*—With the rapid growth of Web services, recommending suitable services to users has become a big challenge. The existing service recommendation works by Quality of Service (QoS) prediction fail to fully consider the influence of situation information, such as time, location, and user relations thoroughly. Two issues must be resolved to consider situation information: issue one, rating scarcity, is that there are less data to learn when importing more situations; issue two is that an effective approach is needed to adapt many situational factors. Our solution is a two-phase method: first, to overcome rating scarcity, data is augmented with estimations of unknown QoS values by learning from observable factors. The augmented data is then used to learn the important latent factors associated with the situational factors for QoS prediction. Experiments on data of real service invocations in different situations show improvement of our method in terms of QoS prediction accuracy over several existing methods, especially in the severe rating scarcity condition. In addition, analysis on parameter selection of proposed method can further assist in obtaining better QoS prediction in practical use.**

***Keywords*-service recommendation; QoS; situation; rating scarcity**

I. Introduction

With the rapid development of the Internet industry, the number of Web services is exploding. Since there are so many services that offer equivalent or similar functionalities for a certain task, it is a challenge to recommend the best services to users. Quality of Services (QoS) attributes are often used to evaluate service performance and quality for service recommendation. QoS attributes include the important metrics used to evaluate service performance (e.g. response time), and are widely employed in describing non-functional properties of Web services for optimizing Web service composition [1], [2]. Various QoS prediction methods have been proposed for service recommendation, including neighborhood-based collaborative filtering methods and model-based techniques [2]–[5].

Classical service recommendation assumes the QoS attributes remain constant over different situations [5]–[7]. Unfortunately, this is not true in the real world. For example, users may experience higher failure rate than usual when invoking services at rush hour. It is hard to accurately estimate QoS values of services if we cannot capture these dynamic characteristics. Some recent studies consider the impact of time on QoS prediction [2], [4]. But using only time is insufficient. We need to consider situation information of different aspects for service recommendation in more complex scenarios.

One significant issue with introducing situation information is rating scarcity, which is also a challenge for classical service recommendation. In real world scenarios there are usually too many services for a user to invoke, so it is impossible to get QoS rating data for all user-service invocations [4], [8]. Attempting to use situation information will aggravate this problem: the more situations introduced, the scarcer QoS data becomes for each situation. Severe rating scarcity will render many existing QoS prediction methods less effective [9]. Rating scarcity need to be controlled to prevent prediction accuracy from degrading.

Another issue is how to handle situation information. Situation is a complex and abstract concept. Many situational factors, such as time, location and relations of users, are important aspects of situation. QoS performance are correlated with the different aspects of situation. Existing recommendation methods focus on the interactions of users and services [5]–[7]. For situated service recommendation, we need to find an effective approach that can be adapted to interactions of users, services, and many situational factors.

In order to tackle these two issues, we further investigate QoS and situations. There are many factors that can affect QoS performance, such as user network condition and service busy hours. QoS can be predicted given the information of these factors. Some factors are observable, and others are latent. In some popular QoS prediction methods, important latent factors of user-service interactions are learned for better prediction accuracies [3], [6]. In addition, many factors are situational. So we make an assumption that latent factors may be correlated with, and thus can be learned from the observable situational factors.

Based on the investigation, we propose a two-phase method which makes good use of both observable factors and latent factors. In phase 1, some unknown QoS values are estimated by random forest regression with several observable factors, such as workflow of services, which is a kind of side information. These estimated QoS values are then used to augment the density of the original QoS data. In phase 2, different situational factors are used to model the situation. Then a latent factor model, tensor factorization is

2474-2473/18/$31.00 ©2018 IEEE
DOI 10.1109/SCC.2018.00025

applied to the user-service-situation tensor of the augmented data [10]. By doing so, we can learn the latent factors for predicting QoS using situation information. Lastly, the effectiveness of the proposed method is evaluated on a QoS dataset of service invocations in different situations gathered from a real-world service platform operated by us.

The rest of this paper is organized as follows: Section II gives a motivating example. A formal definition of the problem are made in Sect. III. In Sect. IV we describe our proposal and Sect. V shows the results of an experiment and evaluation. We discuss some related work in Sect. VI and draw a conclusion in Sect. VII.

II. Motivating example

To describe the problem comprehensively, we give here an example of users in different situations using Web services on a service platform. Users invoke services in different timeslots from different locations, and the platform will record QoS performance for each service invocation.

Assume that there are five users in two different locations. They use Web services to do some tasks, for example, to translate some messages into another language in different timeslots. Some translation services are available, such as Google Translate service (ser1), J-Server Translate service (ser2) and Bing Translator service (ser3). The service platform providing the services records the QoS performance for each service invocation. Observations from previous works on QoS data of various Web services have found that situational factors such as time and location are strongly correlated with QoS performance (e.g. response time) [11], [12]. Thus we also assume that the response time attribute of services in this example will vary with different invocation timeslots and user locations.

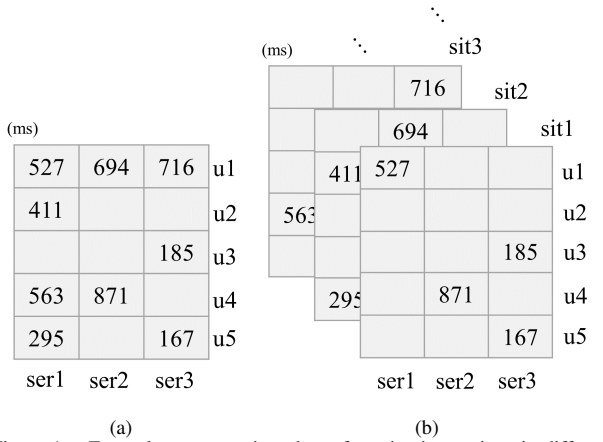

Figure 1. Example response time data of service invocations in different situations (a) The user-service matrix (b) The user-service-situation tensor

QoS data are usually cast into a matrix-like representation for service recommendation. If we assume identical service performance in different situations, we will put the response time values in a user-service matrix as in Fig. 1a. If we consider the situations, which consist of timeslots and locations here, the response times can be put in a user-service-situation tensor as in Fig. 1b. With the tensor, a low response time of ser3 in situation1 (sit1) can be inferred from the known values of user3 and user5 and then ser3 can be recommended for user1 in sit1. But such a recommendation cannot be made if situation information is not considered.

However, using more situation information will aggravate rating scarcity. Comparing Fig. 1a with Fig. 1b, it is clear that the user-service-situation tensor is much scarcer than the corresponding user-service matrix. We need to find ways to compensate the scarcity of data, if we are to make more effective QoS predictions.

As stated in Sect. I, many observable and latent factors can affect QoS performance. Some typical QoS attributes and factors are shown in Fig. 2. Note that many of these factors are situational. Given the availability of situation information from observable situational factors, we can use the latent factor model to capture the dynamic characteristics of latent factors and make accurate QoS predictions.

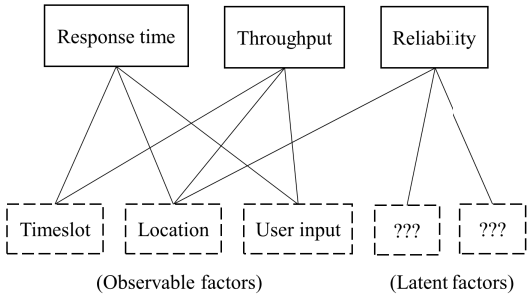

Figure 2. QoS and factors

Therefore, we propose a situated service recommendation method that integrates situation information into the QoS prediction task while addressing the rating scarcity problem raised by introducing situation information.

III. Problem Definition

This section starts by describing and discussing the concept of situation. We then formally define the situated service recommendation problem.

A. Situation and situational factors

A general definition of the word "situation" in the Longman Dictionary is "a combination of all the things that are happening and all the conditions that exist at a particular time in a particular place." In the computing literature, a review of situation identification techniques in pervasive computing defines "situation" as "an external interpretation of sensor data [13]." In a survey of context aware computing, "context," a synonym of "situation," is considered relevant to the interaction between a user and an application [14]. Based on these definitions and the background of our research problem, we define "situation" in this work as:

Definition 1: Situation is the combination of all observable situational factors that are related to the QoS when a service is invoked by a user. Situational factors include time, location and data observed or recorded by sensors or smart devices.

Although latent situational factors may also exist, we do not include them in this definition because we are unable to acquire situation information from them directly.

Formally a situation c is represented by:

$$c = g(f^1, f^2, \dots), \tag{1}$$

where $f^1, f^2, \dots$ are the observable situational factors.

According to the definition, situation is interpreted from all observable situational factors including time, location and sensor data. For convenience, we assume the factors $f^1, f^2, \dots$ are sorted based on their importance. In practice, we choose a few general but important factors based on the scenario to model the situation.

B. Situated service recommendation

The main task for the situated service recommendation problem is to recommend a list of services for each user in a certain situation. This is done by predicting the QoS values of the unknown invocations of user-service-situation interactions.

Consider the example in Sect. II: Given the users, services and situation information, we can record the QoS data in a user-service-situation tensor as in Fig. 1b. Here:

- U is the set of m service users to whom Web services are recommended;
- S is the set of l Web services that are recommended;
- C is the set of p situations as defined in Sect. III-A;
- R is the set of QoS values of user-service invocations in known situations.

Users, services and situations are represented by user ID $i \in \{1, \dots, m\}$, service ID $j \in \{1, \dots, l\}$, and situation ID $k \in \{1, \dots, p\}$ respectively, and:

- $\boldsymbol{Y}$ is the user-service-situation QoS tensor where each entry $\boldsymbol{Y}_{ijk}$ represents the QoS values $r \in R$ of user i, service j in situation k;
- Ω is the set of all tuples of $(user, service, situation)$;
- Λ is the set of all tuples (i, j, k) of the known user-service invocations in known situations, thus $\Lambda \subseteq \Omega$.

We can now formally define the situated service recommendation problem as follows:

Definition 2: Given users U, services S, situations C and QoS values R of the known user-service invocations in situations $(i, j, k) \in \Lambda$, construct a user-service-tensor $\boldsymbol{Y}$ and predict the unknown QoS value entries $\{\boldsymbol{Y}_{ijk} | (i, j, k) \in \Omega - \Lambda\}$ based on the known entries $\{\boldsymbol{Y}_{i'j'k'} | (i', j', k') \in \Lambda\}$.

IV. Proposed method

In order to improve the QoS prediction performance by utilizing different situational factors and resolving rating scarcity issue simultaneously, we propose a new prediction method, Situated Tensor Factorization with Augmented Data (STFAD).

The method consists of two phases: in phase 1, observable factor learning, unknown QoS values are estimated by using several observable factors, and some of these estimated values are augmented to the original QoS data to overcome rating scarcity; in phase 2, latent factor learning, situation is modeled with different situational factors and then tensor factorization is applied to the user-service-situation tensor of the augmented data. Figure 3 briefly shows how our proposal works.

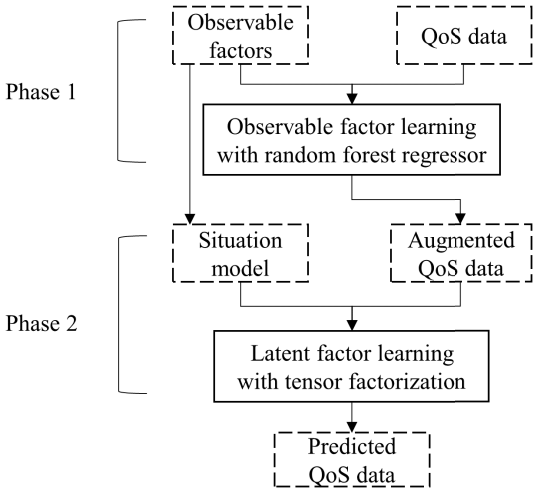

Figure 3. Proposed two-phase method

A. Observable factor learning

Most existing service recommendation methods inevitably suffer a drop in prediction accuracy when the data becomes sparse. Our solution is to enhance QoS data density for the latent factor learning phase by using estimated QoS values to replace the unknown values. This estimation uses some observable factors.

Many machine learning models can be used for this estimation. Here we choose random forest regression [15]. It accepts the QoS values and observable factors as input, and builds multiple randomized decision trees that yield the estimations. We choose this method for two reasons. First, it provides an importance ranking of the input factors; Second, it offers good performance even with large numbers of features, so it will be easy to generalize this method to data with many observable factors.

The random forest regression, $\mathbf{RF}$, learns and predicts unknown QoS values based on known QoS values and observed values of n important observable factors:

$$R' = \mathbf{RF}.learn\_predict(A, R), \tag{2}$$

where A is the set of n important observable factors to learn from. These factors may include some situational factors

and side information that is not used to model the situation. n most important factors are selected from all observable factors. Correlations of observed QoS values and observable factors are learned and used for estimations.

We add some estimated data $R^X \subseteq R'$ to the original QoS data R in order to increase the tensor density for the next phase of latent factor learning. To describe the ratio of estimated data R^X in final QoS data $R^X + R$, we introduce a parameter $\alpha = \frac{|R^X|}{|R^X+R|}$, where $|\cdot|$ is the number of values in the set. A bigger α means more estimated QoS data are augmented to the data tensor for the following latent factor learning phase. As an important parameter that can affect the final QoS prediction performance, its impact will be analyzed in the evaluation section.

B. Latent factor learning

In order to learn the latent factors behind users, services, and situations, we employ tensor factorization to fit a latent factor model to the user-service-situation tensor. The idea behind is to derive representations of the latent factors of users, services and situations by analyzing the user-service-situation tensor. It assumes a low-rank assumption for the QoS tensor, that is, only a small number of latent factors would determine the QoS values by affecting the users, services and situations [3]. Factorization results, in the form of matrices of user-factor, service-factor and situation-factor vectors, are believed to represent the characteristics of physical but latent factors.

There are two popular approaches for tensor factorization: Tucker decomposition and CP decomposition [10]. Here we focus on the CP factorization, where the three component factor matrices share same d latent factors:

$$\boldsymbol{Y} \approx \boldsymbol{I} \times \hat{\boldsymbol{Y}} = \boldsymbol{I} \times \boldsymbol{U} \times \boldsymbol{S} \times \boldsymbol{C}. \tag{3}$$

Here $\boldsymbol{U} \in \mathbb{R}_{m\times d}$, $\boldsymbol{S} \in \mathbb{R}_{l\times d}$, $\boldsymbol{C} \in \mathbb{R}_{p\times d}$ are latent factor matrices (different from sets U, S, C), and $\boldsymbol{I} \in \mathbb{R}_{m\times l\times p}$ is the indicator tensor for $\boldsymbol{Y}$:

$$\boldsymbol{I} = \begin{cases} 1, & \text{if } \boldsymbol{Y}_{ijk} > 0; \\ 0, & \text{otherwise.} \end{cases}$$

The multiplication is defined by:

$$\hat{\boldsymbol{Y}}_{ijk} = \sum_d (\boldsymbol{U}_{id} * \boldsymbol{S}_{jd} * \boldsymbol{C}_{kd}). \tag{4}$$

The columns of $\boldsymbol{U}$, $\boldsymbol{S}$ and $\boldsymbol{C}$ represent user, service and situation, respectively.

With the augmented data tensor $\boldsymbol{X} + \boldsymbol{Y}$ in which $\boldsymbol{X}$ is the corresponding QoS tensor of R^X, we should regulate the latent factor matrices by factorizing the estimated part $\boldsymbol{X}$ simultaneously:

$$\boldsymbol{X} \approx \boldsymbol{I}^{(\boldsymbol{X})} \times \hat{\boldsymbol{Z}} = \boldsymbol{I}^{(\boldsymbol{X})} \times \boldsymbol{U} \times \boldsymbol{S} \times \boldsymbol{C}, \tag{5}$$

where $\boldsymbol{I}^{(\boldsymbol{X})}$ is the indicator tensor for $\boldsymbol{X}$, and $\hat{\boldsymbol{Z}}$ is the predicted QoS tensor on $\boldsymbol{X} + \boldsymbol{Y}$.

To estimate the quality of tensor approximation, a loss function for evaluating the error between the estimated tensor and the original tensor need to be constructed:

$$\begin{aligned} \mathcal{L} =& \frac{1}{2} \sum_{i,j,k} \boldsymbol{I}_{ijk} \|\boldsymbol{Y}_{ijk} - \hat{\boldsymbol{Z}}_{ijk}\|_F^2 + \\ & \frac{\lambda_1}{2} \sum_{i,j,k} \boldsymbol{I}^{(\boldsymbol{X})}_{ijk} \|\boldsymbol{X}_{ijk} - \hat{\boldsymbol{Z}}_{ijk}\|_F^2 + \\ & \frac{\lambda_2}{2} (\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{S}\|_F^2 + \|\boldsymbol{C}\|_F^2), \end{aligned} \tag{6}$$

where the first term measures the sum of errors between observed QoS values and corresponding predicted values, $\|\cdot\|_F^2$ denotes the Frobenius norm, $\|\boldsymbol{U}\|_F^2$,$\|\boldsymbol{S}\|_F^2$ and $\|\boldsymbol{C}\|_F^2$ are three regularization terms to avoid overfitting, $0 \leq \lambda_1 \leq 1$ is a parameter to adjust the influence of augmented data, and λ_2 is a regularization parameter.

The tensor factorization problem can thus be regarded as the optimization problem of minimizing the loss function: $\min_{i,j,k} \mathcal{L}$.

There are several strategies to solve this optimization problem, and here we optimize the loss function by updating latent factor matrices $\boldsymbol{U}$, $\boldsymbol{S}$, $\boldsymbol{C}$ with gradient descent:

$$\boldsymbol{U'} = \boldsymbol{U} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}}, \tag{7a}$$

$$\boldsymbol{S'} = \boldsymbol{S} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}}, \tag{7b}$$

$$\boldsymbol{C'} = \boldsymbol{C} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{C}}. \tag{7c}$$

Finally, we get the updating rules for $\boldsymbol{U}$, $\boldsymbol{S}$, $\boldsymbol{C}$:

$$\begin{aligned} \boldsymbol{U}'_{id} =& \boldsymbol{U}_{id} - [\boldsymbol{I}_{ijk}(\hat{\boldsymbol{Z}}_{ijk} - \boldsymbol{Y}_{ijk})\boldsymbol{S}_j\boldsymbol{C}_k + \\ & \lambda_1 \boldsymbol{I}^{(\boldsymbol{X})}_{ijk}(\hat{Y}_{ijk} - \boldsymbol{X}_{ijk})\boldsymbol{S}_j\boldsymbol{C}_k + \lambda_2 \boldsymbol{U}_{id}], \end{aligned} \tag{8a}$$

$$\begin{aligned} \boldsymbol{S}'_{jd} =& \boldsymbol{S}_{jd} - [\boldsymbol{I}_{ijk}(\hat{\boldsymbol{Z}}_{ijk} - \boldsymbol{Y}_{ijk})\boldsymbol{U}_i\boldsymbol{C}_k + \\ & \lambda_1 \boldsymbol{I}^{(\boldsymbol{X})}_{ijk}(\hat{\boldsymbol{Z}}_{ijk} - \boldsymbol{X}_{ijk})\boldsymbol{U}_i\boldsymbol{C}_k + \lambda_2 S_{jd}], \end{aligned} \tag{8b}$$

$$\begin{aligned} \boldsymbol{C}'_{kd} =& \boldsymbol{C}_{kd} - [\boldsymbol{I}_{ijk}(\hat{\boldsymbol{Z}}_{ijk} - \boldsymbol{Y}_{ijk})\boldsymbol{U}_i\boldsymbol{S}_j + \\ & \lambda_1 \boldsymbol{I}^{(\boldsymbol{X})}_{ijk}(\hat{\boldsymbol{Z}}_{ijk} - \boldsymbol{X}_{ijk})\boldsymbol{U}_i\boldsymbol{S}_j + \lambda_2 \boldsymbol{C}_{kd}] \end{aligned} \tag{8c}$$

for each (i, j, k) and factor d.

Tensor factorization methods may not achieve adequate performance if data density is low. However, given our augmentation of data density in phase 1, good performance is assured. A description of our proposed method in pseudo code is shown in Algorithm 1.

V. EXPERIMENT AND EVALUATION

We conduct an experiment on QoS data of real services collected through an online service platform: the Language Grid[1]. This section compares the proposed method with some existing service recommendation methods on the collected data. Then the impact of some important parameters is

[1]The Language Grid is a service platform of Web services in language domain [16]. It is a project operated by us and allows atomic services to be freely shared and combined to create composite services. Shared or created services are invoked by users around the world, and QoS attributes are recorded for each service invocation. We have done several service computing studies with the Language Grid platform [8], [11], [17].

Algorithm 1: Algorithm for proposed method

Input: QoS values for each service invocation $\{\vec{r}_{ijk}|(i,j,k) \in \Lambda\}$, users $\boldsymbol{U}$, services $\boldsymbol{S}$, situations $\boldsymbol{C}$, the set of n important observable factors A

Output: Predicted user-service-situation QoS tensor $\hat{\boldsymbol{Z}}$

1: Apply random forest regression to predict unknown QoS by eq. (2);
2: Construct tensor $\boldsymbol{Y}$ on $\{\vec{r}_{ijk}|(i,j,k) \in \Lambda\}$, and $\boldsymbol{X} \leftarrow selectAugmentedData(\boldsymbol{R'}, \alpha)$;
3: Initialize factor matrices $\boldsymbol{U} \in \mathbb{R}_{m\times d}$, $\boldsymbol{S} \in \mathbb{R}_{l\times d}$, $\boldsymbol{C} \in \mathbb{R}_{p\times d}$ of user-factor, service-factor and situation-factor vectors;
4: Apply tensor factorization and get predicted tensor $\hat{\boldsymbol{Z}} \leftarrow TF(\boldsymbol{Y}, \boldsymbol{U}, \boldsymbol{S}, \boldsymbol{C}, \boldsymbol{X}, \lambda_1, \lambda_2)$ by applying updating eq. (8) until the error is lower than a given threshold;

analyzed so as to assist in selecting proper ones in practical use to achieve good prediction performance.

A. Experiment settings

We use the Language Grid platform to collect QoS data. Although there are widely-used QoS datasets for QoS prediction, e.g. WS-DREAM [12], they do not provide different situational factors in a single dataset, and so are not suitable for the situated service recommendation problem. Therefore, we choose to use a newly collected set of data that include multiple situational factors.

The experiment settings are as follows.

- **Users**: There are 50 users in total, each has a machine with Internet access to invoke Web services;
- **Services**: Users invoke 232 language services on the Language Grid, most of which are composite services of three or four atomic services;
- **Situational factors**: Two important situational factors are considered for describing a situation:
  - **Time**: Users invoke the services in 12 continuous two-hour timeslots;
  - **Location**: Users are distributed in 5 locations around the world: two in the U.S., one in Japan, one in Australia and one in the U.K.;

  By combining time factor and location factor as the "situation," we have 60 different situations;
- **Data tensor**: The data is recorded in a user-service-situation tensor with size of $50 \times 232 \times 60$;
- **Observable factors**: Nine observable factors are used for the phase 1 of observable factor learning, including factors of time and location of the service invocation, factors that describe the workflow and components of composite services, as well as the factor for describing the length of messages input for translation;
- **Parameters**: Parameter α is set properly for each training data tensor density in the observable factor learning phase to ensure that the final density of augmented QoS data $\boldsymbol{X}+\boldsymbol{Y}$ is 20%. For the latent factor learning phase, λ_1 is set to 1 so that the weight of augmented QoS data equals that of original QoS data. λ_2 is set to 0.02 to control the speed of updating factor matrices;

In this work, only time and location factors are used as situational factors for simplicity and convenience of data collection. Without loss of generality, our method can be extended to more observable situational factors.

The data contains 178,126 valid service invocation records. The response time value ranges from 0–20,000 milliseconds, with an average of around 4,600 milliseconds, as shown in Fig. 4. The response time is relatively large, since we mostly invoke composite services. Because the response time values are highly skewed and within a wide range, a box-cox transformation $bc(r) = \frac{r^{\beta}-1}{r}$ and a normalization $norm(r) = \frac{bc(r)-bc(r)|_{min}}{bc(r)|_{max}-bc(r)|_{min}}$ are performed for each $r \in R$ (here $\beta = 0.25$ is a parameter to maximize the likelihood estimation of the box-cox transformation), which is similar to Zhu et al.'s QoS prediction work [3]. Through these transformations the original response time values are mapped into a normal distribution-like data in range [0, 1].

Tensor data density of the collected data is 26%. We control the data density for training from 1% to 20% by randomly selecting part of the QoS data entries in the tensor, and use the unselected QoS data as the test dataset.

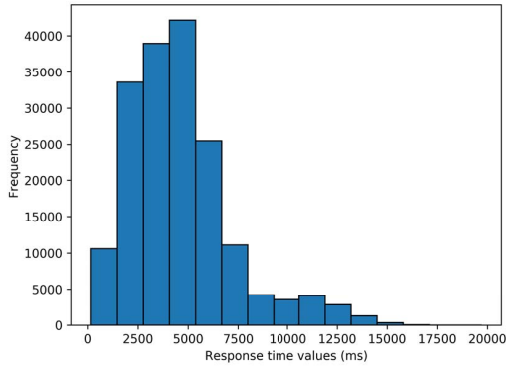

Figure 4. Response time value distribution of collected data

B. Evaluation

1) Evaluation setup: We evaluate the prediction accuracy of our proposed method against other baseline methods with two metrics, MAE and RMSE, which are commonly used in service recommendation studies [2], [4]–[6].

- MAE(Mean Absolute Error):
$$MAE = \frac{1}{N}\sum_{i,j,k}|\boldsymbol{Y'}_{ijk} - \hat{\boldsymbol{Y}}_{ijk}|;$$
- RMSE (Rooted Mean Squared Error):
$$RMSE = \sqrt{\frac{1}{N}\sum_{i,j,k}(\boldsymbol{Y'}_{ijk} - \hat{\boldsymbol{Y}}_{ijk})^2}.$$

Table I
PREDICTION ERROR COMPARISON (A SMALLER VALUE MEANS BETTER PERFORMANCE)

Metric	Method	Training Data Density							
		1%	2%	3%	4%	5%	10%	15%	20%
MAE	WSRec	0.8287	0.7948	0.7676	0.7375	0.7050	0.5534	0.5083	0.5019
	PMF	0.3009	0.1665	0.1383	0.1059	0.0869	0.0710	0.0658	0.0580
	NNCP	0.0681	0.0581	0.0541	0.0533	0.0532	0.0532	0.0545	0.0558
	STF	0.0757	0.0633	0.0579	0.0564	0.0545	0.0529	0.0522	0.0521
	STFAD	**0.0539**	**0.0526**	**0.0520**	**0.0516**	**0.0516**	**0.0519**	**0.0518**	**0.0517**
RMSE	WSRec	0.8415	0.8089	0.7813	0.7515	0.7197	0.5826	0.5561	0.5652
	PMF	0.3612	0.2264	0.2017	0.1563	0.1255	0.0997	0.0925	0.0838
	NNCP	0.0940	0.0827	0.0786	0.0780	0.0780	0.0786	0.0803	0.0819
	STF	0.1038	0.0897	0.0839	0.0824	0.0806	0.0792	0.0783	0.0784
	STFAD	**0.0812**	**0.0798**	**0.0792**	**0.0790**	**0.0787**	**0.0786**	**0.0782**	**0.0781**

Here $\boldsymbol{Y}'_{ijk}$ is the QoS value of Web service s_j observed by user u_i in situation c_k of the test dataset, $\hat{\boldsymbol{Y}}_{ijk}$ denotes the predicted QoS value of Web service s_j by user u_i in situation c_k, and N is the total number of predicted QoS values in testing set.

2) Performance Comparison: Methods used in the comparison are:

- WSRec, aka UIPCC (User-Item based Pearson Correlation Coefficient) [7]: a neighborhood-based collaborative filtering methodwhich calculates the prediction values by considering both user similarity and service (item) similarity;
- PMF (Probabilistic Matrix Factorization) [18]: a widely used implementation of the matrix factorization model. We use this as a baseline of the latent factor method for the user-service QoS matrix;
- NNCP (Non-Negative CP decomposition) [2]: a method where time information is the third dimension of the tensor (other potential situational factors are ignored);
- STF (Situated Tensor Factorization): the phase 2 of our proposed method. Tensor factorization is applied to the user-service-situation tensor. The rating scarcity issue is not considered;
- STFAD (Situated Tensor Factorization with Augmented Data): our proposed method which includes more situation information while addressing rating scarcity.

The comparison of response time prediction accuracy in Table I shows that:

- Our proposed method STFAD consistently attains smaller MAE and RMSE. This shows better prediction accuracy for our approach;
- When training data density is larger than 10%, the prediction errors for STF are smaller than those of the other three existing methods. This indicates a prediction improvement caused by more situational factors;
- However, when training data density is lower than 5%, STF has larger MAE and RMSE values than NNCP. This shows a degradation of prediction accuracy under severe rating scarcity;
- Even when training data density is very low, proposed STFAD still attains smaller errors than the three existing methods. For example, it offers a 14% improvement in RMSE and a 21% improvement in MAE over NNCP under 1% training data density. This indicates that rating scarcity is alleviated by our QoS data augmentation.

In a word, proposed method STFAD outperforms the existing methods as it not only has less error, but also is more stable against low data density because of our consideration of situation information and strategies against rating scarcity.

C. Analysis on important parameters

As can be easily deduced, the number of augmented QoS data controlled by α is an important parameter for the proposed method, especially if training data density is low. The number of most important observable factors and number of latent factors (denoted as dimensionality) are also important parameters, for they affect the prediction performance of phase 1 and phase 2, respectively.

This subsection analyzes on these parameters to elucidate how to select parameters for good prediction performance.

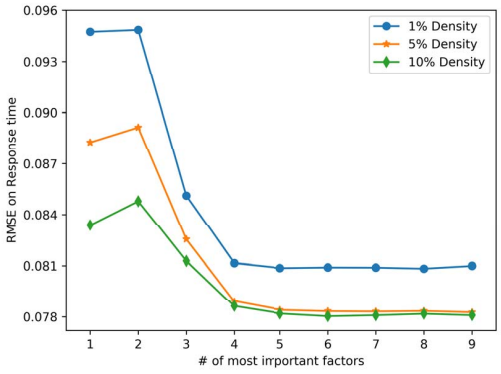

Figure 5. Impact of most important observable factors

1) Impact of most important observable factors: As the random forest regression in phase 1 provides us with a

ranking of the input observable factor importance, we change the number of most important factors to study the impact of these factors on the RMSE values of QoS prediction.

Figure 5 shows that performance generally improves with the number of factors, but the improvement becomes stable when more than four important factors are used. This is because the random forest regression has enough information to make an approximate estimation of unknown QoS values. We can also see that the performance improves with training data density.

2) Impact of augmented QoS data ratio: The augmented QoS data ratio represents the percentage of estimated QoS values is used for the density augmentation, which is the parameter $\alpha = \frac{|R^X|}{|R^X+R|}$ introduced in Sect. IV-A. To study the impact of the augmented QoS data ratio α, we vary it from 0–50% under different training data densities and compare the RMSE values for QoS prediction.

Figure 6 shows that the prediction accuracy generally improves and gradually tends to become stable as α grows. Comparing the results of different training data densities, it can be seen that performance of high-density data becomes stable with small α and overfitting may occur when α is larger, while the prediction accuracy of 1% training data density can still improve when α is near 0.5, which means a large amount of augmented QoS data. This infers that severe rating scarcity is the main reason for poor prediction accuracy. When the density of the tensor with data augmentation reaches a threshold, the influence of rating scarcity is compensated and data augmentation will not provide much prediction improvement.

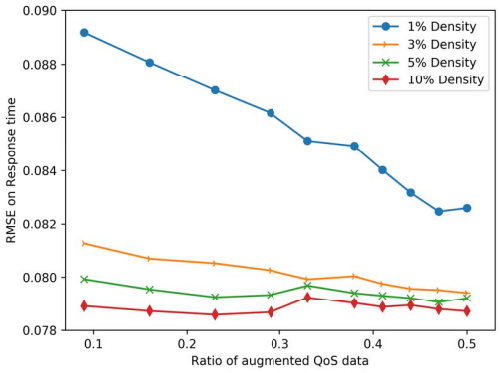

Figure 6. Impact of augmented data ratio

3) Impact of dimensionality: Dimensionality parameter d means the number of latent factors involved for tensor factorization. The impact of dimensionality is investigated by varying the number of latent factors for tensor factorization from 1 to 30 under different input densities.

Previous studies have observed that the prediction accuracy will grow with dimensionality [2], [5]. When dimensionality is too large, overfitting may occur and affect the performance. Similar results can be seen in Fig. 7 for large training data density, but for scarce cases with density less

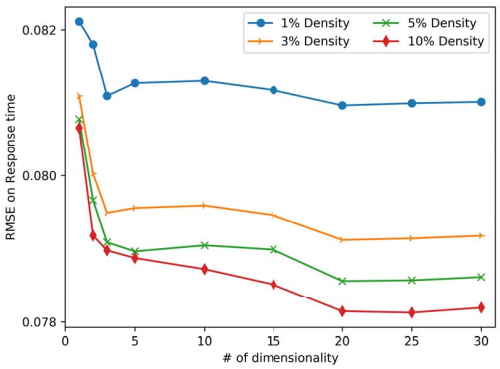

Figure 7. Impact of dimensionality

than 3%, prediction accuracy will degrade occur even for small numbers. This may be caused by premature overfitting of the low training data densities. However, the impact of dimensionality does not seem large in the experiment, as shown by the small changes of accuracy in Fig. 7. Since more factors also incur greater computational complexity, there is no need to choose large dimensionality numbers.

VI. Related Work

Recommender systems have been widely discussed and applied in many problems. Service recommendation is one kind. In this section, we conduct a literature review on Web service recommendation and recommender systems that address the rating scarcity issue.

For the Web service recommendation problem, user-side QoS values are usually regarded as ratings of service invocations. Zheng et al. propose a neighborhood-based method called WSRec: user similarity and item (service) similarity are considered in predicting QoS values of unknown invocations [7]. Zheng et al. also propose a model-based method where user similarity is first calculated before applying matrix factorization [5]. These works however, do not consider the dynamic characteristics of significant factors in different situations. Hu et al. extend WSRec by considering time-based similarity for similarity measurement [4]. Zhang et al. try to include a time factor as the third dimension in the tensor factorization model-based method for Web service recommendation [2]. Although time, clearly an important situational factor, has been discussed in these recent studies, we argue that a single factor cannot describe complex situations perfectly, and thus cannot predict the QoS correctly. To better capture the dynamic characteristics of the QoS, we take more situational factors into consideration and apply tensor factorization to find out user-service-situation correlations.

Rating scarcity is also a challenge in the field of recommender systems. Hu et al. apply a random-walk step for inferring indirect user / service similarities in an effort to alleviate rating scarcity [4]. This issue is studied further in some recommender systems that include additional information. Yao et al. point out that replacing applications

with detailed application versions will make the data matrix sparser and existing methods less effective [9]. They use some extra correlations such as review texts and application similarity. Nakatsuji considers semantic relations to resolve the sparsity problem [19]. However, in our research of the situated service recommendation problem, rating scarcity is aggravated because the observed QoS values are now distributed over different situations within the user-service-situation data tensor. Strategies used in other works, such as similarity and semantic relations, are hard to define and apply to the situated scenarios. We utilize estimations on unknown QoS data entries by learning from observable factors and use these estimated QoS data to augment the data density to solve this problem.

VII. Conclusion

Situation information should be considered for Web service recommendation because the QoS performance is highly correlated with situations. Yet considering more situation information aggravates rating scarcity, which makes existing QoS prediction methods for recommendation less effective. In addition, an effective approach is needed to adapt many situational factors. In order to resolve the two issues, we propose a two-phase method: first, QoS data are augmented with estimations of some unknown QoS values by learning from observable factors; then latent factors are learned on the user-service-situation tensor of augmented data for predicting QoS. The performance of the proposed method is evaluated by an experiment on QoS data of real service invocations in different situations. Comparison with existing methods shows that the proposal offers better prediction performance, especially with scarce input data. We also investigate the impact of different important parameters, which can help us select proper parameters for better prediction performance in practical use.

Although our proposed method has shown good performance on QoS data with two situational factors, for a more general situation model of many factors, more work need to done considering the increasing rating scarcity. In future work, we would like to consider more situational factors and try to develop a better situation model. Preprocessing strategies on situational factors to further decrease the loss of data density, such as clustering, will also be considered.

Acknowledgment

This research was partially supported by a Grant-in-Aid for Scientific Research (A) (17H00759, 2017-2020) and a Grant-in-Aid for Scientific Research (B) (18H03341, 2018-2020) from Japan Society for the Promotion of Science (JSPS), and the Kyoto University Foundation.

References

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

[2] 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*. ACM, 2014, pp. 585–596.

[3] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online qos prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2911–2924, 2017.

[4] Y. Hu, Q. Peng, X. Hu, and R. Yang, "Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering," *IEEE Trans. Services Comput.*, vol. 8, no. 5, pp. 782–794, 2015.

[5] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service qos prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 289–299, 2013.

[6] J. Xu, Z. Zheng, and M. R. Lyu, "Web service personalized quality of service prediction via reputation-based matrix factorization," *IEEE Trans. Rel.*, vol. 65, no. 1, pp. 28–37, 2016.

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

[8] X. Zhou, D. Lin, and T. Ishida, "Evaluating reputation of web services under rating scarcity," in *2016 IEEE International Conference on Services Computing (SCC)*. IEEE, 2016, pp. 211–218.

[9] Y. Yao, W. X. Zhao, Y. Wang, H. Tong, F. Xu, and J. Lu, "Version-aware rating prediction for mobile app recommendation," *ACM Trans. Inf. Syst.*, vol. 35, no. 4, pp. 38:1–38:33, 2017.

[10] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[11] D. Lin, C. Shi, and T. Ishida, "Dynamic service selection based on context-aware qos," in *2012 IEEE International Conference on Services Computing (SCC)*. IEEE, 2012, pp. 641–648.

[12] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating qos of real-world web services," *IEEE Trans. Services Comput.*, vol. 7, no. 1, pp. 32–39, 2014.

[13] J. Ye, S. Dobson, and S. McKeever, "Situation identification techniques in pervasive computing: A review," *Pervasive Mob. Comput.*, vol. 8, no. 1, pp. 36–66, 2012.

[14] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 2014.

[15] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[16] T. Ishida, Ed., *The Language Grid: Service-Oriented Collective Intelligence for Language Resource Interoperability*. Springer, 2011.

[17] Y. Murakami, D. Lin, and T. Ishida, Eds., *Services Computing for Language Resources*. Springer, 2018.

[18] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems 20*. Curran Associates, Inc., 2008, pp. 1257–1264.

[19] M. Nakatsuji, "Semantic sensitive simultaneous tensor factorization," in *The Semantic Web – ISWC 2016*. Springer, 2016, pp. 411–427.