Personalized LSTM Based Matrix Factorization for Online QoS Prediction

Ruibin Xiong, Jian Wang, Zhongqiao Li, Bing Li
School of Computer Science,
Wuhan University,
Wuhan, China
Email: {ruibinxiong, jianwang, zhongqiaoli, bingli}@whu.edu.cn

Patrick C. K. Hung
Faculty of Business and IT,
University of Ontario Institute of Technology,
Canada
Email: patrick.hung@uoit.ca

***Abstract*—Quality of Service (QoS) prediction is an important task in services computing, which has been extensively investigated in the past decade. Many time-aware QoS prediction approaches have been proposed and achieved encouraging prediction performance. However, they did not provide effective model updating mechanisms, and thus have to periodically retrain the whole models to deal with the newly coming data. How to timely update the prediction model to precisely predict missing QoS values of candidate services becomes an urgent issue. In this paper, we propose a novel personalized LSTM based matrix factorization approach for online QoS prediction. Our approach can capture the dynamic latent representations of multiple users and services, and the prediction model can be timely updated to deal with the new data. Experiments conducted on a real-world dataset show that our approach outperforms several state-of-the-art approaches in online prediction performance.**

***Index Terms*—Web service; QoS prediction; time-aware; online model; personalized LSTM.**

I. INTRODUCTION

With the wide adoption of service-oriented computing, an increasing number of software applications have been developed by reusing and composing Web services to deliver desired functionalities, which in turn speeds up the creation of Web services or Web APIs. The growing number of Web services brings a heavy burden for developers to select suitable services in building service-based systems. The ultimate objective of service selection is to find services that can meet both functional and non-functional requirements of users. With the growing supply of functionally similar services, the non-functional properties of services thus become major considerations in decision making of service selection.

Quality of Service (QoS) refers to the non-functional aspects of Web services, such as response time, throughput and cost. It serves as a basis to determine which services could be selected or recommended. Generally speaking, QoS values can be measured at both provider-side and user-side [1], [2]. Due to the dynamic network environment and different geographical locations of service users, different users may observe totally different QoS values when they invoke the same service. More attention should be paid to the user-side QoS evaluations, which are more relevant to personalized user experiences on Web services. The straightforward way to obtain the user-side QoS values is to perform real invocations by users, which is infeasible in practice due to the time-consuming process and expensive overhead [3]. Therefore, how to accurately predict unknown QoS values of services for different users becomes an important issue in services computing.

In recent years, many approaches [2], [4], [5] have been proposed to predict unknown QoS values of Web services according to their historical invocation records. Considering the high volatilities of QoS values over time, many time-aware approaches [1], [3], [6]–[9] have been proposed predicting QoS values according to the service requesting time of target users, which have obtained encouraging prediction performance. In the real-world QoS prediction scenario, new QoS observations are coming continuously as time goes by. Whether the trained prediction models need to be updated and how to update them have not been sufficiently discussed and addressed in existing literature. Towards this end, we firstly conducted empirical experiments to analyze performances of existing time-aware approaches on model updating. The dataset used in the experiments is a variant of WSDream [2], the most popular QoS prediction dataset, which consists of QoS data of 4500 services observed by 140 users over 64 time intervals. More descriptions of the dataset, experimental settings, and evaluation metrics (MAE and RMSE) can be found in Section V.A. Here we just illustrate some observations on two representative time-aware QoS prediction approaches: CLUS [4] and WSPred [5].

- From Table I, we can observe that the updated models using the latest observations outperform the original ones that only leverage historical observations. Therefore, **more recent QoS observations will result in more accurate prediction performance**, which further confirms the hypothesis proposed in [6].
- From Table II, we can observe that these two time-aware methods, which leverage historical data for training, outperform a classic non-time-aware method, which only leverages the latest QoS data. Furthermore, the average improved percentage over intervals 44-63 (where more historical data are accumulated for training) is better than that over intervals 1-20. Therefore, **more QoS observations used in the model training will result in more accurate prediction performance.**

978-1-5386-7247-1/18/$31.00 ©2018 IEEE
DOI 10.1109/ICWS.2018.00012

TABLE I: PERFOMANCE IMPROVEMENT BY MODEL UPDATING

Method	Matrix density = 5%		Matrix density = 10%		Matrix density = 15%		Matrix density = 20%	
	Decreased percentage on MAE	Decreased percentage on RMSE	Decreased percentage on MAE	Decreased percentage on RMSE	Decreased percentage on MAE	Decreased percentage on RMSE	Decreased percentage on MAE	Decreased percentage on RMSE
CLUS	3.16%	3.58%	7.89%	4.23%	9.95%	6.12%	11.42%	7.30%
WSPred	4.93%	3.57%	5.21%	3.93%	6.95%	4.99%	7.89%	5.48%

Note: We trained models using the data of the first 10 (i.e., intervals 0-9), 20, 30, and 40 time intervals, denoted as original models, as well as those of the last 10 (i.e., intervals 54-63), 20 30, and 40 time intervals, denoted as updated models. The latest QoS observations (i.e., the data of interval 63) are used for prediction. Each value in the table indicates average decreased percentage on MAE and RMSE (i.e., performance improvement) by using original models and updated ones.

TABLE II: PERFORMANCE IMPROVEMENT BY DIFFERENT SIZE OF HISTORICAL DATA

Method	Average improved percent over intervals 1-20		Average improved percent over intervals 44-63	
	MAE	RMSE	MAE	RMSE
CLUS	3.65%	3.33%	8.69%	5.59%
WSPred	8.50%	13.14%	17.51%	20.85%

Note: We compared the two time-aware prediction methods with a classic non-time-aware method NMF [10] over time intervals 1-63. Interval 1 means that QoS values of intervals 0 and 1 are used for training, while interval 63 indicates that QoS values of intervals 0 to 63 are used.

According to the conclusions shown in the experiments, the prediction model needs to be updated incrementally by adding the newly coming data to obtain better prediction performance. However, most existing time-aware approaches seldom provide effective model updating mechanisms. They have to retrain the model using large amounts of historical data, which makes them difficult to meet the requirements of online prediction. Therefore, how to timely update the prediction model to provide online prediction becomes an important issue. To address this challenge, in this paper, we propose a novel personalized LSTM (Long Short-Term Memory) based matrix factorization approach (PLMF, for short) for online QoS prediction. Our model can characterize the dynamic latent representations of multiple users and services and can be updated according to the current observations and limited historical data, along with some long-term retained information.

The main contributions of the paper are summarized as follows:

- We study the problem of online QoS prediction of Web services, which has not been sufficiently discussed in existing literature.
- We propose a novel personalized LSTM based matrix factorization approach for online QoS prediction.
- Experiments conducted on a real-world dataset show that our approach outperforms several state-of-the-art QoS prediction models in online prediction performance.

The rest of the paper is organized as follows. Section II discusses the related work. Preliminaries are introduced in Section III. The proposed online QoS prediction approach is introduced in Section IV. Experimental results and analysis are given in Section V. Finally, Section VI summarizes the paper and puts forward our future work.

II. RELATED WORK

As a critical technique for Web service recommendation and selection in dynamic environments, QoS prediction has been investigated in depth in the past decade. Existing studies towards this problem can be classified into two types according to whether they can deal with time-series QoS data.

The first group of studies in this area mainly focused on personalized QoS prediction at a static time interval, where the past QoS observations are not considered. Collaborative filtering (CF) approaches are the most widely applied techniques in this scenario, which can be approximately classified into two types: neighborhood-based and model-based. Neighborhood-based approaches measure similarities of users or services according to their past QoS experiences and then leverage QoS experiences of similar neighbors to make personalized QoS prediction for the user on candidate Web services. The neighborhood-based CF approaches can be further classified into user-based, service-based and hybrid approaches [2]. Model-based approaches employ machine learning techniques to learn models based on the training data for future prediction. Matrix factorization (MF) is the representative of this type, e.g., CloudPred [11] and NMF [10]. Existing literature has shown that MF-based approaches are better than neighborhood-based ones on prediction performance. However, both of these two kinds of approaches cannot capture the temporal dynamics of QoS experiences, which hinders them from obtaining ideal results.

Another group of studies, time-aware QoS prediction approaches, train models to fit the past QoS values and then predict their future trends. For example, Godse et al. [12] proposed to predict service performance based on the ARIMA (Auto Regressive Integrated Moving Average) model. Amin et al. [13] integrated ARIMA with GARCH models in QoS prediction. However, ARIMA based models mainly focus on predicting QoS values for each individual Web service observed by one user, which makes them intractable in predicting multiple services observed by multiple users during service selection and personalized recommendation. CF is thus introduced to alleviate the issue. For example, Hu et al. [14] proposed a prediction model by combining CF with improved time series forecasting based on ARIMA and Kalman filtering. The works [5], [15] formalized the time-aware QoS prediction

problem using a user-service-time tensor and performed tensor factorization. Silic et al. [4] proposed to cluster historical QoS data and hash the average reliability values of each cluster for prediction. Zhu et al. adopted a similar strategy and proposed a novel context-specific matrix factorization method [3]. As analyzed in Section I, most existing time-aware approaches did not provide effective model updating mechanisms. They have to periodically retrain their models to deal with the new data. Inspired by the idea of recurrent recommender networks [16] and collaborative RNN [17], we propose a new personalized LSTM model, which can capture temporal dependencies of both users and services so as to predict QoS in an online manner.

In addition, Wang et al. [17] applied the classical LSTM in online reliability prediction for a single service observed by a single user. In contrast, our work can predict QoS of multiple services for multiple users simultaneously, which is thus more suitable to meet the needs of service selection. Zhang et al. [8] proposed a QoS forecasting approach based on multivariate time series, which presents a feasible way of leveraging the correlations among multiple QoS attributes. As for the model updating mechanism, their model updates parameters according to the latest multivariate QoS attributes of a single service, while ours can collaboratively update the prediction model according to the QoS attributes of multiple services observed by multiple users.

III. PRELIMINARIES

In this section, we briefly introduce some preliminary knowledge to be used in this paper.

A. Matrix Factorization

As mentioned in Section II, Matrix factorization (MF) based techniques have been widely used in Web service QoS prediction. Given a user-service QoS observation matrix $\boldsymbol{R} \in \mathbb{R}^{U \times S}$ from U users and S services, MF learns user latent representations $p_u \in \mathbb{R}^K (u = 1, ..., M)$ and service latent representations $q_s \in \mathbb{R}^K (s = 1, ..., N)$ from $\boldsymbol{R}$, and estimates QoS values $r_{us} \subseteq \boldsymbol{R}$ as Equation (1).

$$\hat{r}_{us} = F\left(p_u, q_s | \Theta\right), \tag{1}$$

where F denotes the interaction function of p_u and q_s, and Θ denotes the parameters of F. Usually, F can be an inner product of two vectors p_u and q_s.

B. Long Short-Term Memory

Long short-term memory (LSTM) [18] is a kind of variants of the recurrent neural network (RNN), which can capture temporal dynamics from the sequence. Compared to classic RNN, LSTM introduces a gated unit consisting of input gate i_t, forget gate f_t, output gate o_t and memory cell state c_t. At each time interval t, a new input x_t along with the latest unit output $h_{(t-1)}$ will be transformed and accumulated into the cell state if the input gate i_t is activated. Also, the past cell state can be forgotten by forget gate f_t. Then, the output gate o_t will determine what information of c_t can be output.

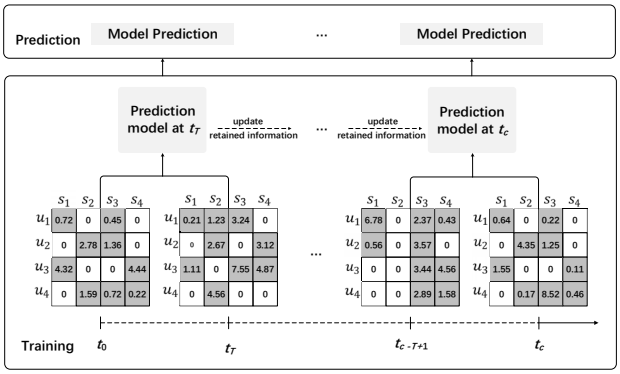


Fig. 1: Overall framework

In the QoS prediction scenario, the classic LSTM model can be used to learn a sequence float pattern as done in [17]. However, it cannot capture potential differences between different users or different services, which result in a poor performance, as mentioned before. Therefore, we propose a novel personalized LSTM model for QoS prediction.

IV. OUR APPROACH

A. Overall Framework

1) Problem statement: Suppose that there are M users $U = \{u_1, u_2, ..., u_M\}$ and N Web services $S = \{s_1, s_2, ..., s_N\}$. When a user invokes a Web service, he (or she) can observe a QoS value of the service from his (or her) own perspective. In this way, each service has varied QoS values observed by different users, and these values may change over time. If user u_i $(u_i \in U)$ invoked service s_j $(s_j \in S)$ at time interval t_k $(k = 1, 2, ...)$, the QoS value of this service invocation will be recorded. Given a service request of user u_i at current time interval t_c, the problem to be addressed in this paper is how to timely and precisely predict missing QoS values of candidate services for service selection and recommendation.

2) Basic solution: As discussed in Section I, most existing approaches cannot incrementally incorporate the latest QoS observations into their prediction models, which hinders them from providing ideal online prediction performance. In this paper, we aim to propose an online QoS prediction approach to address the issue. Fig. 1 shows the framework of the proposed approach.

In the training stage, our model tries to capture the temporal dependencies of both users and services. As shown in Fig. 1, t_c denotes the current time interval, and t_{c-T+1} denotes the $T-1$-th time interval ahead of t_c. At t_c, our approach leverages the current observation $\boldsymbol{R}^{(t_c)}$, some historical data $\boldsymbol{R}^{(t_c-T+1)}, \boldsymbol{R}^{(t_c-T+2)}, ..., \boldsymbol{R}^{(t_c-1)}$ within $T-1$ time intervals, along with the long-term retained information, to update the latest model. In the next time interval, t_c becomes $t_{(c-1)}$ and the whole calculation moves forward by one time interval, which can be viewed as a sliding window with length T. The model will be periodically updated with the new data as time goes by. In the prediction stage, given a service request of user

u_i, personalized QoS prediction for all candidate services can be calculated according to the latest trained model.

B. Analysis of online QoS prediction using MF

Since matrix factorization has shown its clear advantages in QoS prediction, we also attempt to adapt the idea of MF in the online QoS prediction scenario. Following the MF paradigm, the QoS value of service s invoked by user u at time interval t can be estimated by the combination of latent representations of u and s, as shown in Equation (2).

$$\hat{r}_{us}^{(t)} = F(p_u^{(t)}, q_s^{(t)}|\Theta), \tag{2}$$

where $p_u^{(t)}$ and $q_s^{(t)}$ denote the latent representations of user u and service s at time interval t, respectively, F denotes the interaction function, and Θ denotes parameters of F. Let $\boldsymbol{P}^{(t)} \in \mathbb{R}^{M\times K}$ and $\boldsymbol{Q}^{(t)} \in \mathbb{R}^{N\times K}$ denote the latent representation matrices of users and services, respectively. If the classic MF technique is adopted, the calculation of $\boldsymbol{P}^{(t)}$ and $\boldsymbol{Q}^{(t)}$ solely depends on $\boldsymbol{R}^{(t)}$, the QoS observations at t. However, this strategy cannot make full use of the past observations nor can it capture temporal dependencies, which will thus result in poor prediction performances. Thanks to the advantages of LSTM in capturing temporal dependencies, we use two LSTMs to learn $\boldsymbol{P}^{(t)}$ and $\boldsymbol{Q}^{(t)}$, respectively, to overcome these mentioned limitations.

C. Personalized LSTM

To characterize the dynamic representations of multiple users and services in QoS prediction, we introduced the aspect of collaborative filtering into the classical LSTM model and propose a novel personalized LSTM model (P-LSTM), which can capture dynamic latent representation for multiple users and services.

As depicted in Fig. 2, the key difference between P-LSTM and the classical LSTM lies in that, the weight matrices of forget gate f_t, input gate i_t, output gate o_t and internal state c_t are represented as embedding matrices, which can be viewed as embedding lookup tables of users and services. Their outputs can be viewed as dense representations of users or services, akin to latent factors in matrix factorization. In this way, these structures become personalized, whose outputs vary by user or service. $f_t^{[u]}$, $i_t^{[u]}$, $o_t^{[u]}$ and $c_t^{[u]}$ denote the corresponding personalized structures of user u. Take forget gate $f_t^{[u]}$ as an example, the embedding matrices of $f_t^{[u]}$ can be represented as:

$$\boldsymbol{W}_f^{[t]} = [\boldsymbol{W}_f^{[t][1]}, \boldsymbol{W}_f^{[t][1]}, ..., \boldsymbol{W}_f^{[t][M]}], \tag{3}$$

where $\boldsymbol{W}_f^{[t][u]} \in \mathbb{R}^K$ (K denotes the size of an embedding), M denotes the number of users, and $\boldsymbol{W}_f^{[t][u]}$ denotes the forget gate embedding for user u. When user u is input, its forget gate embedding $\boldsymbol{W}_f^{[t][u]}$ will be output and participated in the calculation of $f_t^{[u]}$ along with the latest output $h_{t-1}^{[u]}$. In essence, $f_t^{[u]}$ describes what information of $c_{t-1}^{[u]}$ is forgotten.

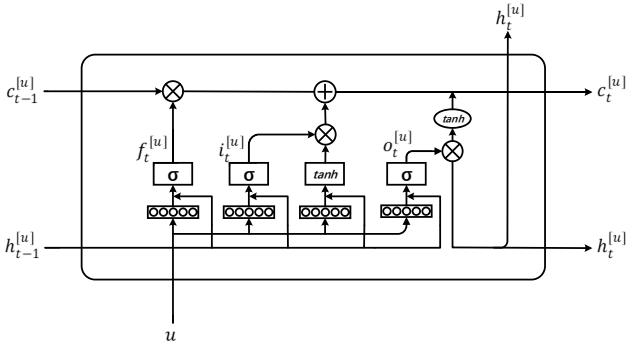


Fig. 2: Structure of P-LSTM

Formally, the calculation process of the personalized LSTM for user u can be described as:

$$\begin{aligned}
f_t^{[u]} &= \sigma(\boldsymbol{W}_f^{[t][u]} + \boldsymbol{V}_f h_{t-1} + b_f),\\
i_t^{[u]} &= \sigma(\boldsymbol{W}_i^{[t][u]} + \boldsymbol{V}_i h_{t-1} + b_i),\\
o_t^{[u]} &= \sigma(\boldsymbol{W}_o^{[t][u]} + \boldsymbol{V}_o h_{t-1} + b_o),\\
\tilde{c}_t^{[u]} &= i_t^{[u]} \odot tanh(\boldsymbol{W}_c^{[t][u]} + \boldsymbol{V}_c h_{t-1} + b_c),\\
c_t^{[u]} &= f_t^{[u]} \odot c_{t-1}^{[u]} + \tilde{c}_t^{[u]},\\
h_t^{[u]} &= o_t \odot tanh(c_t^{[u]}),
\end{aligned} \tag{4}$$

where $h_t^{[u]}$ denotes the output of personalized LSTM for u at time interval t, $c_t^{[t]}$ denotes the cell state, $\boldsymbol{W}$ and $\boldsymbol{V}$ denote weight matrices, b denotes a bias term, σ denotes the sigmoid function, and $\odot$ denotes the Hadamard product. For service s, the calculation process is similar.

D. Personalized LSTM based Matrix Factorization

1) Model Training: The structure of PLMF is depicted in Fig. 3. Note that at each time interval, PLMF uses observations of T intervals in the sliding window along with the "long term memory" to update the latest model. Given user u and service s, the model will estimate $\left[\hat{r}_{us}^{t_c-T+1}, ..., \hat{r}_{us}^{t_c-1}, \hat{r}_{us}^{t_c}\right]$ for training. We firstly focus on the estimation of $\hat{r}_{us}^{t}$ at interval t, and expand it to the whole estimation process within the sliding window, followed by the optimization of the model. At the top input layer, the identifiers of u and s are first transformed into sparse binary vectors with one-hot encodings (for example, $[0, 1, 0, 0, ..., 0, 0]$), and then fed into two P-LSTMs: a user-side P-LSTM and a service-side P-LSTM, respectively, to learn dynamic latent representations. Let $p_u^{(t-1)}$ and $q_s^{(t-1)}$ denote latent representations of u and s, respectively; $c_{t-1}^{[u]}$ and $c_{t-1}^{[s]}$ denote internal states of u and s, respectively. The dynamic latent representations can be estimated as follows.

$$\begin{aligned}
p_u^{(t)}, c_t^{[u]} &= P - LSTM_{user}(u, p_u^{(t-1)}, c_t^{[u]}),\\
q_s^{(t)}, c_t^{[s]} &= P - LSTM_{service}(s, q_s^{(t-1)}, c_t^{[s]}),
\end{aligned} \tag{5}$$

where $P - LSTM_{user}$ and $P - LSTM_{service}$ denote the calculation process of the user-side and the service-side, respectively. Once the latent representations of users and

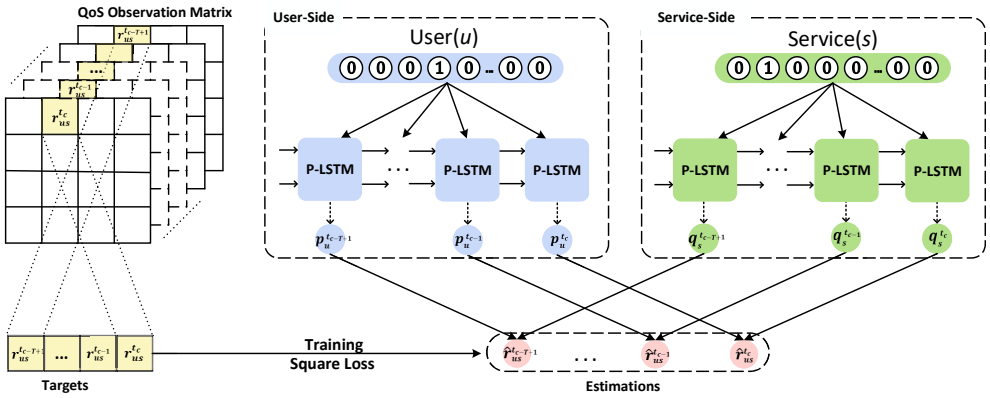


Fig. 3: Structure of PLMF

services are obtained, we can follow the matrix factorization paradigm to estimate QoS values, as shown in Equation (6).

$$\hat{r}_{us}^{(t)} = f(p_u^{(t)}, q_s^{(t)}) = \sum_{k=1}^{K} p_{uk}^{(t)} q_{sk}^{(t)}, \tag{6}$$

where K denotes the dimension of latent representation vectors. Please note that we can also adopt other nonlinear mappings like [19]. Considering the requirement of computation complexity, we adopt the linear combination of p_u^t and q_s^t to estimate $\hat{r}_{us}^{(t)}$.

To capture more reliable temporal dependencies, the historical QoS data can be incorporated into the model training of each interval. As mentioned before, there is a sliding window with size T across the timeline. At each new time interval, the sliding window moves forward by one interval, and the QoS observations within the window are collected as training set $\mathcal{D}_{train}^{(t_c)}$ for model updating. The model estimates $\left[\hat{r}_{us}^{t_c-T+1}, ..., \hat{r}_{us}^{t_c-1}, \hat{r}_{us}^{t_c}\right]$ according to Equations (6) and (7).

$$\left.\begin{aligned} p_u^{(t_c-T+1)}, c_{t_c-T+1}^{[u]} &= P-LSTM_{user}(u, p_u^{(t_c-T)}, c_{t_c-T}^{[u]}), \\ p_u^{(t_c-T+2)}, c_{t_c-T+2}^{[u]} &= P-LSTM_{user}(u, p_u^{(t_c-T+1)}, c_{t_c-T+1}^{[u]}), \\ &\cdots \\ p_u^{(t_c)}, c_{t_c}^{[u]} &= P-LSTM_{user}(u, p_u^{(t_c-1)}, c_{t_c-1}^{[u]}), \end{aligned}\right\} T$$
$$\left.\begin{aligned} q_s^{(t_c-T+1)}, c_{t_c-T+1}^{[u]} &= P-LSTM_{service}(s, p_s^{(t_c-T)}, c_{t_c-T}^{[s]}), \\ p_s^{(t_c-T+2)}, c_{t_c-T+2}^{[u]} &= P-LSTM_{service}(s, p_s^{(t_c-T+1)}, c_{t_c-T+1}^{[s]}), \\ &\cdots \\ p_s^{(t_c)}, c_{t_c}^{[s]} &= P-LSTM_{service}(u, p_s^{(t_c-1)}, c_{t_c-1}^{[s]}), \end{aligned}\right\} T \tag{7}$$

where $p_u^{t_c-T}$ and $q_s^{t_c-T}$ are dynamic latent representations that need to be hold by PLMF at t_c. We then predict $\hat{r}_{us}^{t_c}$ using $p_u^{t_c}$ and $q_s^{t_c}$ according to Equation (6).

Under this setting, the optimization objective is to find parameters that yield predictions close to the actual QoS observations across T time intervals. We adopt mean square error (MSE) as the loss function and use L_2 regularization to control the model complexity. The cost function $\mathcal{L}$ to be minimized is defined as follows.

$$\mathcal{L} = \sum_{u,s} \sum_{t=t_c-T+1}^{t_c} I_{ust}^{(t_c)} (r_{us}^{(t)} - \hat{r}_{us}^{(t)})^2 + \frac{\lambda}{2} ||\Theta||^2, \tag{8}$$

where $I_{ust}^{(t_c)}$ is an indicator function: $I_{ust}^{(t_c)} = 1$ if $r_{us}^{(t)} > 0$ in $D_{train}^{(t_c)}$, and $I_{ust}^{(t_c)} = 0$, otherwise. Θ denotes the parameters of two P-LSTMs, and λ denotes the regularization parameters.

BPTT (Backpropagation Through Time) has been widely used in existing literature [17], [20] to train the LSTM model. However, it is not suitable to directly apply BPTT in our problem, because QoS predictions depend on two P-LSTMs simultaneously. To alleviate this issue, we first fix the user-side parameters, and traverse the training set to update the P-LSTM of the service-side; then we fix the service-side parameters and traverse the training set to update the user-side. Using this strategy, we can update the user-side P-LSTM and the service-side P-LSTM alternatively, until the model converges or reaches the early-stopping condition. Moreover, we use Adaptive Moment Estimation (Adam) to optimize our proposed model.

2) Prediction: Using the trained PLMF model at t_c, we can calculate dynamic latent representations for u and s according to Equation (7), and further estimate QoS value $\hat{r}_{us}^{(t)}$ according to Equation (6). Due to the non-negative nature of the QoS, the negative prediction is adjusted to zero.

V. EXPERIMENTS

In this section, we conducted a series of experiments to evaluate the proposed method. These experiments were designed to answer the following research questions:

- **RQ1** Does the proposed method outperform the state-of-the-art QoS prediction approaches in terms of prediction performance?
- **RQ2** Is the updating time of our model shorter than existing time-aware QoS prediction approaches?
- **RQ3** Can the QoS prediction performance be improved by increasing the size of the sliding window?

A. Experimental settings

All experiment programs were carried out on a GPU Workstation with Intel Core 8 CPU Xeon(R) E5-1620 v3, @3.50 GHz, GeForce GTX 1080 and 32 GB RAM, running the Ubuntu 16.04 OS. We implemented our approach based on Pytorch[1], a widely adopted deep learning framework, and ran it on the GPU. For competing methods, we adopted their publicly available versions[2], implemented in C++.

1) Dataset: WSDream [2], a real-world Web service QoS dataset, is the most popular dataset in QoS prediction. We created a new dataset based on the raw observations of WSDream by removing users with extremely sparse data and adding side-information (e.g., locations and IPs) of users and services for future follow-up studies. The new dataset consists of the response time of 4500 services observed by 140 users over 64 time intervals, for a total of 27,392,643 records. The sparsity is about 66.98%. The source code and the dataset used in this paper can be retrieved at our project page[3].

2) Evalaution Metrics: We adopt Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to measure prediction performance. MAE is defined as:

$$MAE = \frac{1}{|\mathcal{D}_{test}|} \sum_{(u,s,t)\in\mathcal{D}_{test}} \left| r_{us}^{(t)} - \hat{r}_{us}^{(t)} \right|, \tag{9}$$

where $r_{us}^{(t)}$ denotes the actual QoS value of service s observed by user u at time interval t, $\hat{r}_{us}^{(t)}$ denotes the predicted QoS value of s observed by u at t, and $|\mathcal{D}_{test}|$ represents the size of testing set $\mathcal{D}_{test}$. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{|\mathcal{D}_{test}|} \sum_{(u,s,t)\in\mathcal{D}_{test}} (r_{us}^{(t)} - \hat{r}_{us}^{(t)})^2}. \tag{10}$$

3) Competing approaches: To evaluate the effectiveness of our approach, we chose several state-of-the-art QoS prediction approaches for comparison. Among these approaches, CloudPred [11] and NMF [10] are non-time-aware, while WSPred [5], CARP [3], and CLUS [4] are time-aware. Since these competing approaches are not natural online models, in order to perform a fair comparison with our PLMF, we enable these approaches to leverage the latest QoS data. The detailed strategy is described as follows. Suppose at time interval t, the historical data and the current ones are denoted as $\boldsymbol{R}^{(1)}, ..., \boldsymbol{R}^{(t)}$. Firstly, QoS data at each interval were split into two disjoint sets: training set and testing set, according to a pre-defined ratio, denoted as $\boldsymbol{R}_{train}^{(1)}, ..., \boldsymbol{R}_{train}^{(t)}$ and $\boldsymbol{R}_{test}^{(1)}, ..., \boldsymbol{R}_{test}^{(t)}$. Then for time-aware methods, $\boldsymbol{R}_{train}^{(1)}, ..., \boldsymbol{R}_{train}^{(t)}$ were chosen as $\mathcal{D}_{train}$ for training, and for non-time-aware methods, only the current observation set $\boldsymbol{R}_{train}^{(t)}$ was chosen as $\mathcal{D}_{train}$. For all approaches, $\boldsymbol{R}_{test}^{(t)}$ was chosen as $\mathcal{D}_{test}$ for testing. The training and testing processes were conducted to evaluate performance at each time interval.

[1]http://pytorch.org/

[2]https://github.com/wsdream

[3]https://github.com/chrisxiongrb/PLMF

4) Parameter Settings: We randomly sampled 10 percent of the training set to construct a cross-validation set and tuned the hyper-parameters accordingly. Model parameters were randomly initialized by following a Gaussian distribution. According to the parameter tuning results, we used the full batch learning and set the learning rate α, regularization parameters λ and dropout ratio in P-LSTM to 0.02, 30.0 and 0.25, respectively. In Section V.B, we set window size T to 3 and the size of latent representation K to 80. In Sections V.C and V.D, we set K to 64, 80, 80, 96 for $T = 1, 2, 3, 4$, respectively. We also found that the better performance can be obtained by feeding the model with pre-trained latent representations. The latent representations were initialized by a pre-trained NMF.

As for the competing methods, we tuned their hyper-parameters by following the default settings in WSDream.

B. Performance Comparison (RQ1)

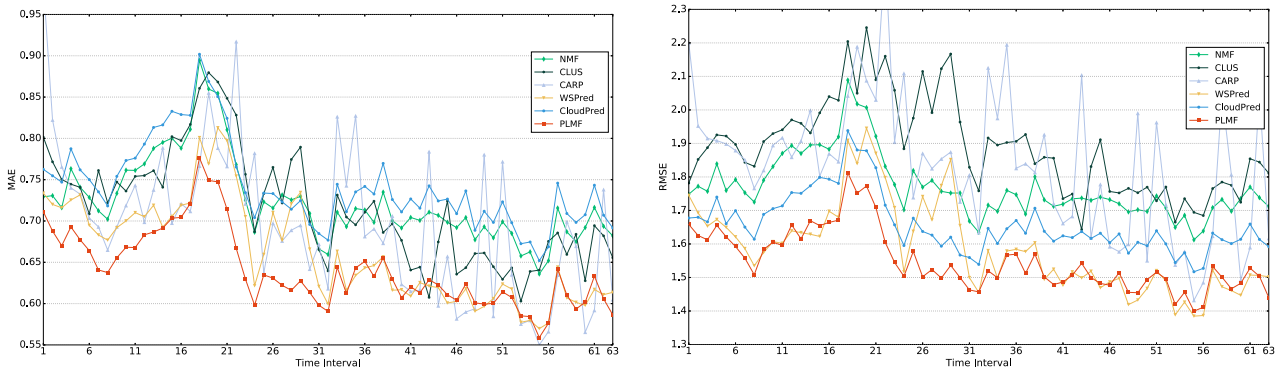
We set different data density to construct a training set from 5% to 20% with a step size 5%. Comparison results are presented in Table III.

Table III provides the comparison result of QoS prediction performance. Note that each value is a weighted average over all intervals. We can observe that our PLMF approach consistently outperforms the other approaches with smaller MAE and RMSE. In particular, PLMF achieves larger performance improvement when the density becomes larger, which demonstrates the representation ability of our approach.

To compare the detailed performance over every interval, Fig. 4 shows the detailed performance of all methods on the training set with density 20% at intervals 1-63. As shown in Fig. 4, the performances of all methods exhibit similar change trends with time intervals, excepting that CARP's performance changes steeper. Compared with non-time-aware approaches NMF and CloudPred, PLMF outperforms them at all intervals. Compared with time-aware approaches, PLMF also achieves the best at most time intervals, especially when the data are relatively scarce. It indicates that classic time-

TABLE III: QOS PREDICTION PERFORMANCE COMPARISON

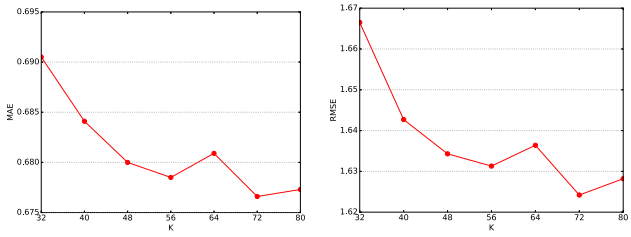
Metrics	Method	Matrix Density			
		5%	10%	15%	20%
MAE	NMF	0.9041	0.8021	0.7548	0.7251
	CloudPred	0.8747	0.8066	0.7694	0.7432
	WSPred	0.7317	0.6894	0.6734	0.6635
	CLUS	0.7842	0.7542	0.7350	0.7185
	CARP	0.8759	0.7709	0.7282	0.6992
	PLMF	**0.7267**	**0.6786**	**0.6582**	**0.6444**
RMSE	NMF	2.2330	1.9853	1.8570	1.7749
	CloudPred	1.8957	1.7751	1.7079	1.6634
	WSPred	1.7074	1.6339	1.6068	1.5931
	CLUS	1.8921	1.9030	1.9046	1.8948
	CARP	2.2437	2.0397	1.9424	1.8556
	PLMF	**1.7059**	**1.6126**	**1.5749**	**1.5525**



(a) MAE (Matrix Density = 20%)

(b) RMSE (Matrix Density = 20%)

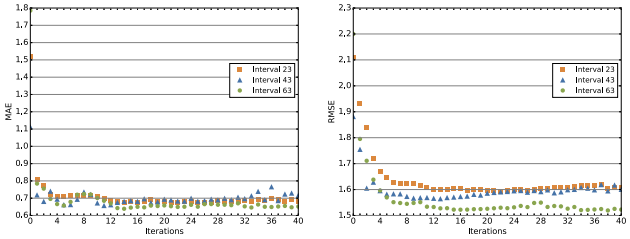
Fig. 4: Performance comparison at every time interval



(a) MAE (Matrix Density = 10%)

(b) RMSE (Matrix Density = 10%)

Fig. 5: Performance of PLMF w.r.t. K



(a) MAE (Matrix Density = 10%)

(b) RMSE (Matrix Density = 10%)

Fig. 6: Performance of PLMF w.r.t. the number of iterations

aware approaches require a large amount of data to achieve relatively better performance. Please note that, at each time interval, the time-aware approaches leverage not only the data of the current interval but also all historical data to train a new model. In the contrast, PLMF only uses the data of the current interval along with historical data of a few time intervals (in this group of experiments, historical data of 2 intervals were used) to update the latest model. For example, at time interval 40, time-aware approaches use all data of 40 intervals (from 1 to 40) to train a new model, while PLMF only uses data of 3 intervals (from 38 to 40). As can be seen in Section V.C, as the amount of training data increases, the training time spent on the model increases dramatically and goes far beyond PLMF.

1) Impact of K: As stated before, PLMF adopts the matrix factorization paradigm for QoS prediction by learning latent representations of users and services, whose sizes are K. Since K controls the model complexity, its setting is a key issue. To study the impact of K on QoS prediction, we set K varying from 32 to 80 with a step size 8. Fig. 5 shows the MAE and RMSE of different K. The result indicates that when K is increased to an appropriate range, the representation ability of the model can be enhanced. While K increases from 32 to 72, both MAE and RMSE values show a nearly upward trend, in spite of some fluctuations happen when $K = 64$. When K exceeds this range, the performance shows a downward trend.

C. Updating Time Comparison (RQ2)

In online learning scenarios, the training speed is another critical issue. To investigate whether PLMF updates faster than other time-aware approaches, we selected three representative time intervals (i.e., 23, 43 and 63), and then studied the convergence of PLMF on the test set over these three intervals. We then compared the training time to achieve optimal performance between PLMF and other methods over these three intervals. The experiment ran 10 times and took the average as the result. Fig. 6 shows the convergence of PLMF over three intervals. We can observe that PLMF can obtain ideal performance within limited iterations. Take interval 43 as an example, the model obtains the best RMSE after 13 iterations. Moreover, within 10 iterations, PLMF obtains 98.3%, 99.9%, 98.4% of the best RMSE, on intervals 23, 43 and 63, respectively. The result of MAE shows the similar trends. Comparison results on the training time are shown in Table IV, where PLMF-1, PLMF-2, and PLMF-3 denote window size $T = 1, 2, 3$ for PLMF, respectively. Because the prediction performances of non-time-aware methods are not comparable to others, we only selected time-aware methods for comparison. Firstly, with the increase of T, the training time shows an increasing trend in spite of some exceptions, e.g., at interval 63, the training time of PLMF-3 is shorter than PLMF-1. The exception can be explained by that although the training time of a single iteration gets longer as T increases, the convergence speed may be faster. Secondly, as the interval increases, the training time of time-aware methods increases dramatically, due to the increase of the amount of training data; in contrast, PLMF maintains good stability in the training time due to its online nature. We can conclude that, in the online learning scenario, as time goes by, PLMF has a significant

TABLE IV: COMPARISON OF TRAINING TIME

Method	Training Time (s)		
	Interval 23	Interval 43	Interval 63
CARP	123.2	168.70	262.34
CLUS	45.20	56.95	221.99
WSPred	356.53	701.39	1081.31
PLMF-1	**31.70**	**16.40**	**47.46**
PLMF-2	**39.33**	**32.49**	**64.98**
PLMF-3	112.11	86.38	**23.31**

TABLE V: PERFORMANCE OF PLMF W.R.T SIZE OF SLIDING WINDOW

Metrics	Size of Sliding Window T			
	1	2	3	4
MAE	0.6809	0.6834	0.6794	0.6689
RMSE	1.6364	1.6204	1.6161	1.6075

advantage in the training speed compared to other methods.

D. Will A Longer Sliding Window Be Helpful? (RQ3)

Using the sliding window, our approach can incorporate more data in each update, but it may also result in longer training time. Theoretically, a longer sliding window can help the model capture more stable temporal dependencies. In this experiment, we want to empirically investigate whether a longer sliding window will contribute to a better performance. We adjusted the size of sliding window T from 1 to 4, while fixing other parameters. The experiment was still conducted on the dataset with matrix density 10%. From Table V, we can observe that when T is set from 1 to 4, both MAE and RMSE show a downward trend. The results indicate that it is possible to further improve prediction performance by increasing the size of the sliding window, which is encouraging.

VI. Conclusion

In this paper, we propose a novel personalized LSTM based matrix factorization approach for online QoS prediction. Our approach can capture the dynamic latent representations of multiple users and services, and the prediction model can be timely updated to deal with newly coming data. In the future, we plan to apply our approach to other real-world datasets that consist of more QoS properties. We also consider integrating more side information such as locations and IP addresses of users and services into our model.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2017YFB1400602), the National Basic Research Program of China (No. 2014CB340404), the National Natural Science Foundation of China (Nos. 61572371, 61672387 and 61702378), and the Natural Science Foundation of Hubei Province of China (Nos. 2018CFB511 and 2017CKB894). The authors are grateful for Prof. Zibin Zheng's help with providing the raw QoS data. Jian Wang is the corresponding author.

References

[1] Y. Wu, F. Xie, L. Chen, C. Chen, and Z. Zheng, "An embedding based factorization machine approach for web service qos prediction," in *International Conference on Service-Oriented Computing*. Springer, 2017, pp. 272–286.

[2] 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.

[3] J. Zhu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Carp: Context-aware reliability prediction of black-box web services," in *Web Services (ICWS), 2017 IEEE International Conference on*. IEEE, 2017, pp. 17–24.

[4] M. Silic, G. Delac, and S. Srbljic, "Prediction of atomic web services reliability for qos-aware recommendation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 425–438, 2015.

[5] Y. Zhang, Z. Zheng, and M. R. Lyu, "Wspred: A time-aware personalized qos prediction framework for web services," in *Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium on*. IEEE, 2011, pp. 210–219.

[6] Y. Hu, Q. Peng, and X. Hu, "A time-aware and data sparsity tolerant approach for web service recommendation," in *Web Services (ICWS), 2014 IEEE International Conference on*. IEEE, 2014, pp. 33–40.

[7] W. Li, P. Zhang, H. Leung, and S. Ji, "A novel qos prediction approach for cloud services using bayesian network model," *IEEE Access*, vol. 6, pp. 1391–1406, 2018.

[8] P. Zhang, L. Wang, W. Li, H. Leung, and W. Song, "A web service qos forecasting approach based on multivariate time series," in *Web Services (ICWS), 2017 IEEE International Conference on*. IEEE, 2017, pp. 146–153.

[9] P. Zhang, H. Jin, Z. He, H. Leung, W. Song, and Y. Jiang, "Igs-wbsrm: A time-aware web service qos monitoring approach in dynamic environments," *Information and Software Technology*, 2017.

[10] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[11] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based qos prediction in cloud computing," in *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on*. IEEE, 2011, pp. 1–10.

[12] M. Godse, U. Bellur, and R. Sonar, "Automating qos based service selection," in *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 534–541.

[13] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting qos attributes of web services based on arima and garch models," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 74–81.

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

[15] Y. Ma, S. Wang, F. Yang, and R. N. Chang, "Predicting qos values via multi-dimensional qos data for web service recommendations," in *Web Services (ICWS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 249–256.

[16] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 495–503.

[17] Y. J. Ko, L. Maystre, and M. Grossglauser, "Collaborative recurrent neural networks for dynamic recommender systems," in *Journal of Machine Learning Research: Workshop and Conference Proceedings*, vol. 63, 2016.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[20] H. Wang, Z. Yang, and Q. Yu, "Online reliability prediction via long short term memory for service-oriented systems," in *Web Services (ICWS), 2017 IEEE International Conference on*. IEEE, 2017, pp. 81–88.