QF-RNN: QI-Matrix Factorization Based RNN for Time-Aware Service Recommendation

Xing Wu, Yushun Fan*
Beijing National Research Center for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 100084, China
wuxing17@mails.tsinghua.edu.cn
fanyus@tsinghua.edu.cn

Jia Zhang
Department of Electrical and Computer Engineering
Carnegie Mellon University
Silicon Valley
Moffett Field
CA 94035, USA
jia.zhang@sv.cmu.edu

Haozhe Lin, Junqi Zhang
Beijing National Research Center for Information Science and Technology
Department of Automation
Tsinghua University
Beijing 100084, China
linhz16@mails.tsinghua.edu.cn
jq-zhang15@mails.tsinghua.edu.cn

***Abstract*—Driven by the widespread application of Service Oriented Architecture (SOA), the quantity of Web services and their users keep increasing in the service ecosystem. If historical service invocation records can be gathered and accumulated, it is meaningful to recommend suitable services that users may invoke in the near future. However, most existing recommend algorithms bear major limitation of not taking consideration of dynamic characteristics of both users and Quality of Service (QoS). To address this concern, this paper proposes a time-aware recommendation algorithm for runtime service selection. Firstly, a QoS Observation Matrix is created integrated with Invocation Record Matrix. Afterwards, matrix factorization is applied to extract user-preferences and service-features, respectively. Due to their dynamic characteristics, the Long Short Term Memory (LSTM) model is leveraged to learn and predict preferences and features. Finally, a service recommendation list is generated for users based on LSTM predictions. Experimental results on a real-world dataset show that the proposed algorithm outperforms baseline methods in terms of accuracy and recall.**

***Index Terms*—service recommendation; matrix factorization; RNN; QoS; time-aware**

I. Introduction

With the extensive adoption of Service Oriented Architecture (SOA) and Cloud Computing in the last decade, numerous services have been published onto the Internet, and their user bases have been dramatically increased as well [1]. Web services can provide electronic tourist guide, online shopping, social intercourse etc. in an efficient and flexible way. Users can also get access to these convenient services comfortably with the help of ubiquitous mobile devices, such as iPhone or iPad [2]. In theory, anyone can enjoy any service at any time in the way they like. But how to select one from the sea of services which can meet user's demand and improve user's experience? That really makes a challenge.

In the field of service oriented computing, many service recommendation algorithms have been proposed to find services which can meet user's functional requirements. For example, if a user needs navigation in an unfamiliar city, most existing algorithms can provide a list of recommendations containing *Baidu Maps* or *Google Maps*. Because either of them can meet user's need, to further recommend the better suitable service, we should take more attention on the non-functional properties of services, which have become the major considerations in the trends of service oriented computing. Quality of Service (QoS) refers to the non-functional aspects of Web services, such as response time, throughput and cost [3] [4]. On this specific issue, if Google Maps has a higher QoS for the specific city, we would choose this service for recommendation. Note that QoS may also impact user's Quality of Experience (QoE). For instance, if a user continues to encounter unsatisfactory experiences from a service, his related preferences may decline to some extent. Therefore, for each service, it is meaningful to pay attention to its QoS value when making recommendations.

In addition to QoS factor in general, to recommend appropriate services that users may actually invoke in the near future, some other non-negligible facts require special attention:

- User's direct functional demands may not always be available. For example, if a user consequently invokes *Senic Spot Search* and *Airline Reservation*, we can reasonably infer that the user is planning a trip. Then services, like *Hotel Accommodation* and *Tourist Guide*, could be recommended. In the era of information and Artificial Intelligence (AI), the key is to mine user's implicit requirements actively instead of waiting for search instructions passively.
- The actual behavior of invoking a service is driven by some invisible features like user's hidden preferences. For example, if a user shows interest in *World Cup* and *Sportradar Olympics*, he may be an enthusiastic sports fan. With the help of such a hidden feature, *Yahoo Sports* or *FIFA* would be appropriate recommendations.
- User's preferences always show dynamic characteristics. On the one hand, user's long-term preferences may drift over time due to personal internal factors. On the other hand, the user's short-term interests may jump by occasional leaps. For example, a wonderful experience of a newborn service is likely to lead to a new preference

immediately. Therefore, the ability to mine long-short preferences is an important property of a recommendation algorithm.

Such non-negligible facts call for new methods. The literature has witnessed a number of approaches to recommend Web services. While many works mainly take functional requirements into consideration [5] [6], most non-functional properties-based works only handle with static problems [10] [11]. Thus they cannot predict the tendency of future invocation. Some time-aware methods consider only dynamic characteristics of user's preferences, however, ignoring the effect of QoS variety on recommendation results [15] [16]. Only few approaches [20] consider both dynamics of user's preferences and QoS variety. However, two problems remain for those approaches. Firstly, they just take the recent QoS into consideration, without mining the dynamic characteristics of QoS. Secondly, they take user's preferences and QoS as two independent variables, while neglecting their latent connection. To overcome the limitations of previous work, in this paper, a novel recommendation algorithm, named QI-Matrix Factorization Based Recurrent Neural Networks (QF-RNN), is proposed.

Firstly, we integrate QoS metrics into invocation records with certain rules to obtain QI-matrix, which can comprehensively reflect whether the service is invoked or not and the QoS value of invoked services. Then at every time slice, the matrix factorization method is used to excavate user's preferences and service's features at different time points. After obtaining preferences and features which both show continuous characteristics, we incorporate Long Short Term Memory (LSTM) into the analysis of time series, to predict preferences and features in the future. Based on the prediction results of LSTM, the QI-matrix of the subsequent moment can be extended, which is the basis of recommending top-N services with the highest QoS.

The main contributions of this paper are summarized as follows.

- We propose a novel QI-matrix factorization-based RNN, which uses matrix factorization to extract complicated characteristics and applies LSTM units to make time-aware service recommendation.
- We fully consider the influence of QoS value on user's dynamic preferences, regarding QoS as an important factor affecting recommendation results instead of one indicator for evaluating recommendation results. To our best knowledge, this is the first attempt to recommend top-N services considering dynamic QoS factors over users.
- Extensive experiments over a real-world dataset WS-Dream show that QF-RNN outperforms baseline methods in terms of prediction accuracy.

The rest of this paper is organized as follows. Section II discusses the related work. The framework of our QF-RNN model is described in Section III. Experimental results and analyses are given in Section VI. Finally, Section V summarizes the paper and puts forward our future work.

II. Related work

Existing service recommendation approaches mainly consider their functional and non-functional features. As for non-functional features, the related works can be further divided into static approaches and time-aware approaches.

A. Functional Approaches

With respect to the functional approaches, the top priority is to recommend services with the most suitable functions to users rather than thinking about their indexes like QoS or ease of use. Some functional approaches exploited content matching like key-words search [5] [6], while some other approaches leveraged semantics-based search to increase the accuracy of the recommendation [7]. Chen et al. [8] proposed a Time-aware Collaborative Poisson Factorization (TCPF), taking poisson factorization as the foundation to model mashup queries and service descriptions separately. Hao et al. [9] reconstructed service description and developed a new service recommendation strategy accordingly.

B. Non-functional Static Approaches

Non-functional service recommendation methods pay more attention to the QoS or service network analysis. Collaborative filtering, which is based on the assumption that similar users tend to invoke similar services, has been widely used in static approaches. Zheng et al. [25] introduced two collaborative filtering based Web service recommendation approaches, neighborhood-based and region-based approaches, to help users select Web service with optimal QoS performance. Tang et al. [10] took location information into consideration and improved the recommendation performance. Tan et al. [11] modelled service usage patterns of an evolving service system. Zhou et al. [12] improved the performance of service ranking by performing services ranking and clustering mutually in a heterogeneous service network. Yu et al. [28] integrated the trace norm as a regularization component into the Non-negative Matrix Tri-Factorization (NMTF) process and developed the Trace Norm Regularized Matrix Factorization (TNR-MF) algorithm for QoS prediction.

In recent years, the Latent Dirichlet Allocation (LDA) [13] probability preference model in the field of machine learning has also been introduced into the field of service recommendation. Gao et al. [14] proposed Seco-LDA to discover meaningful latent service composition including their temporal strength and services'impacts and then make service composition recommendation. Liu et al. [15] proposed an iExpand approach based on the LDA model, which enabled a better understanding of the interactions among users, items and user interests.

These static methods neglect the dynamic change of user's preferences; thus, it is hard to make accurate recommendation based on user's recent preferences.

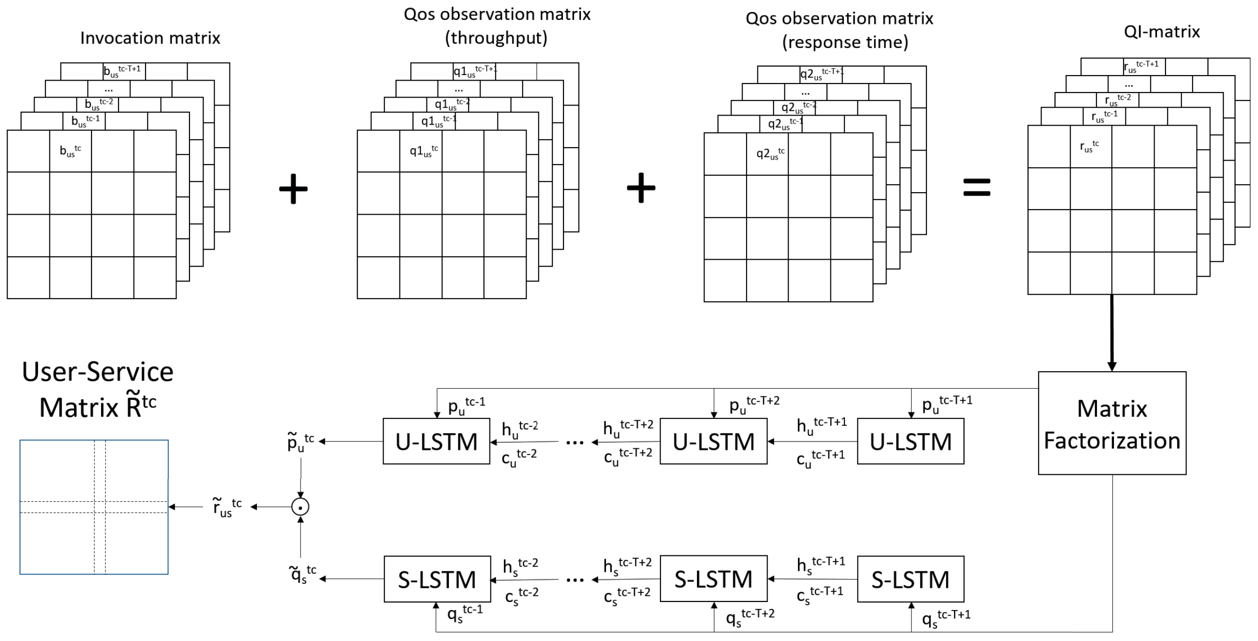

Fig. 1. Structure of QF-RNN. At each time slice, QF-RNN integrates QoS observation matrix into invocation matrix, resulting in QI-matrix. Then matrix factorization is used to extract latent representations. sending them into U-LSTM unit and S-LSTM unit respectively, after sufficient parameter learning, QF-RNN obtains model parameters and can predict the score for each user-service pair at time t_c. QF-RNN returns the score, which is the basis for service recommendation.

C. Non-functional Time-Aware Approaches

The research of service recommendation algorithm considering time information has just arisen in recent years. Yu et al. [29] took time factor into account when computing degree of similarity between services and users and proposed a time-aware collaborative filtering algorithm for QoS-based service recommendation. Hu et al. improved collaborative filtering by integrating time information into both the similarity measurement and the final QoS prediction. Blei et al. [17] developed a dynamic topic model and used it to capture the evolution of topics in sequentially organized corpus of documents.

Some studies take evolution of service usage over time into account while making recommendation. Zhong et al. [18] developed a time-aware service recommendation approach based on LDA, consisting of three components: temporal information, mashup-description-based collaborative filtering and service-description-based content matching. Huang et al. [19] explored the evolution mechanism in the service ecosystem and proposed a three-phased network prediction-based approach for the service and composition recommendation.

While such time-aware studies existing, very few approaches consider both dynamic preferences and dynamic QoS when making recommendation. Zhang et al. [20] extracted user preferences, and made final recommendation by leveraging QoS and matching the interest degree of each user-service pair with corresponding QoS value on the latest time slice. Its limitation is that it just takes the recent QoS into consideration without mining the dynamic characteristics of QoS.

III. Model framework

In this section, we firstly restate the problem mathematically, and then describe the construction of QF-RNN, whose overall framework is depicted in Fig. 1. Finally, we explain how to use the trained model to make recommendation.

A. Notation and Problem Definition

Suppose that there are M users $U = \{u_1, u_2, \cdots, u_M\}$ and N Web services $S = \{s_1, s_2, \cdots s_N\}$. Split total time into C time slices, $T = \{t_1, t_2, \cdots, t_c\}$. At each time slice, we can use invocation matrix $B_t = b_{tij}(1 \leq i \leq M, 1 \leq j \leq N)$ to record invocation records. For example, if user i invokes service j at time interval t, b_{tij} = 1. Besides the invocation matrix, a user can observe a QoS value of the service from his own perspective. QoS usually contains more than one value (such as throughput, response time and cost). Similar to B_t, we use $Q1_t = q1_{tij}$ to represent QoS1 (throughput) of one invocation, $Q2_t = q2_{tij}$ to represent QoS2 (response time).

The main goal of this paper is to precisely recommend what a user will invoke at t_d using historical invocation B_t and historical QoS value Q_t, t from t_1 to t_{d-1}. Furthermore, because services with high QoS values always bring users better experience, if a user actually invokes P services, we shall only recommend N services (N≤P) restricting to some external factors. It is thus meaningful to recommend top-N services with the highest QoS values for users. It is the further goal of this paper.

B. Invocation Matrix and QoS Observation Matrix Integration

As shown in section A, from invocation records and QoS observations, at any time slice t, we can get invocation matrix B_t and QoS matrix $Q1_t$, $Q2_t$,$\cdots$. We then integrate Q_t into B_t under a specific rule, which can be described in Eq. 1.

$$R_t = k_0 \cdot B_t + k_1 \cdot Q1_t + k_2 \cdot \frac{1}{Q2_t} + \cdots \quad (1)$$

where k_0, k_1, k_2 are weight coefficients. We call $R_t = r_{tij}(1 \leq i \leq M, 1 \leq j \leq N)$ as QoS Observation Matrix integrated with Invocation Record Matrix (QI-Matrix). r_{tij} = 0 means user i does not invoke service j at time slice t. If $r_{tij} > 0$, user i invokes service j at time slice t, and we can obtain QoS property intuitively by observing the value of r_{tij}.

C. QI-Matrix Factorization

Matrix Factorization (MF)-based techniques have been widely used in Web service recommendation [23]. Given a user-service QI-matrix $R \in R^{M \times N}$ from M users and N services, we can decompose it into two matrixes — user-preference matrix P, where $p_u \in R_K(u = 1, 2, \cdots M)$ and feature-service matrix Q, where $q_s \in R_K(s = 1, 2, \cdots N)$. The principle of matrix factorization is shown in Fig. 2. The rationale is two-fold. Firstly, we can extract hidden features, which can be set as input of neural network. Secondly, the dimension of matrix can be reduced effectively.

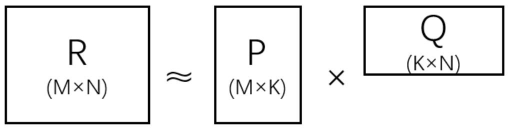

Fig. 2. The principle of matrix factorization. R, P, Q represent QI-matrix, user-preference matrix, and feature-service matrix, respectively.

The solving process of QI-matrix factorization is to estimate $\tilde{r}_{us}$ and make $\tilde{r}_{us}$ as close with r$_{us}$ (the truth QI-matrix value) as possible, which can be carried out using an iterative strategy. The calculation of $\tilde{r}_{us}$ is described in Eq. 2.

$$\tilde{r}_{us} = F(p_u, q_s \mid \Theta) \quad (2)$$

where F denotes the interaction function of p_u and q_s, and Θ denotes the parameters of F. In this paper, F presents the inner product of two vectors p_u and q_s.

D. Long Short-Term Memory

Long Short-Term Memory (LSTM) is an variant of recurrent neural network (RNN), which can capture temporal dynamics from the sequence [21]. Shown in Fig. 3, 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. In our LSTM model, a fully connected (FC) layer is incorporated. 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 is activated. Meanwhile, 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 the current unit output h_t. The above calculation process can be described as Eq. 3 - 8 mathematically.

$$ft = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (3)$$
$$it = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (4)$$
$$\tilde{c}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \quad (5)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (6)$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (7)$$
$$h_t = o_t \odot tanh(c_t) \quad (8)$$

where $\odot$ represents the dot product; W_i, W_f, W_o and W_c are parameters of gates and the memory cell; $\sigma(\cdot)$ and $tanh(\cdot)$ are sigmoid function and hyperbolic function, respectively.

Based on the hidden states h_t, a FC layer is able to decode them into prediction value $\tilde{y}_{t+1}$, which is trained to estimate the truth value y_{t+1}. It can be described in Eq. 9.

$$\tilde{y}_{t+1} = w^{\top} \cdot h_t \quad (9)$$

where w are weight parameters of fully connected layers.

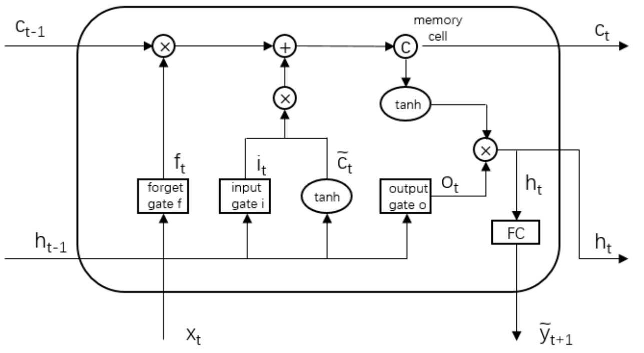

Fig. 3. An LSTM unit in the context of QF-RNN, consisting of input gate i_t, forget gate f_t, output gate o_t and memory cell state c_t. The hidden states h_t will be calculated at each time slice, which will be used as input of both LSTM units and fully connected layers.

Since user's invocation records can be represented as sequences, it is natural to incorporate LSTM into our algorithm. If we simply use user's historical invocation records to predict future invocation, however, we may ignore the effect of user's preferences on recommendation, which results in a poor performance. Therefore, we propose a novel QI-matrix factorization based RNN to further improve the recommendation accuracy.

E. QI-Matrix Factorization Based RNN

1) Model Training: In the training phase, our model tries to excavate the dynamic change rules of user-preferences and service-features. We adopt a learning strategy of sliding windows and set the slider size to T. As described in section C, via the method of matrix factorization, we have obtained user's preferences and service's features at different time point. At user-side, for every user i, we set $[p_i^{t-T+1}, p_i^{t-T+2}, \cdots, p_i^{t-1}]$ as the model input, and set p_i^t as the ground truth, which

represents the real preferences of user i at time t. The model will estimate $\tilde{p}_i^t$ for training. Our data for training come from two sources: on the one hand, select the time interval of length T from the same user at different time t; on the other hand, select from different users. We set i from 1 to M and set t from 64 - T + 1 to 64. Through sufficient iterations, we get the trained user-side LSTM model U-LSTM.

At service-side, a similar deal is taken. For every service j, we set $[q_j^{t-T+1}, q_j^{t-T+2}, \cdots, q_j^{t-1}]$ as the model input, and set q_j^t as the ground truth, which represents the real features of service j at time t. The model will estimate $\tilde{q}_j^t$ for training. We set j from 1 to N and t from 64 - T + 1 to 64. Through sufficient iterations, we get the trained service-side LSTM model S-LSTM.

Under this setting, the optimization objective is to find parameters that yield predictions close to the actual value. We consider adopting mean square error (MSE) as the loss function. To avoid overfitting, we apply ℓ_2 regularization to control the model complexity. The cost function L to be minimized is defined as follows.

$$\begin{aligned} L_{U-LSTM} &= \sum_u (p_u^{tc} - \tilde{p}_u^{tc})^2 + \frac{\lambda}{2}\|\Theta\|^2 \\ L_{S-LSTM} &= \sum_s (q_s^{tc} - \tilde{q}_s^{tc})^2 + \frac{\lambda}{2}\|\Theta\|^2 \end{aligned} \tag{10}$$

Back Propagation Through Time (BPTT) has been widely used in existing literature [22] [23], to train the LSTM model. Moreover, we use Adaptive Moment Estimation (Adam) to optimize our proposed model. We also adopt the gradient clipping method to eliminate the gradient exploding problem [24].

2) Model prediction: Using the trained model, if we want to recommend services for users at time tc , setting $[p_i^{tc-T+1}, p_i^{tc-T+2}, \cdots, p_i^{tc-1}]$ and $[q_j^{tc-T+1}, q_j^{t-T+2}, \cdots, q_j^{tc-1}]$ as the input of U-LSTM and S-LSTM respectively, we can calculate latent representations for u_i and s_j.

F. Recommendation Based on LSTM Predictions

In this paper, we aim to recommend top-N services with the highest QoS for users. Using the trained U-LSTM model, we can calculate $\tilde{p}_i^{t_c}$ for user i. For every j in $[1, N]$, we use the trained S-LSTM model to calculate $\tilde{q}_j^{t_c}$. We also calculate the inner product $\tilde{r}_{ij}$ according to Equation (2), which stands for the score of service j. The Top-N services will be recommended for user i.

The detailed process can be described as follows: Dataset is divided into time slices. At each time slice, we can integrate QoS observation matrix (Q_t) into invocation matrix (B_t), resulting in QI-matrix (R_t). Using matrix factorization to extract latent representations and sending them into the model, after sufficient parameter learning and BPTT algorithm we can obtain model parameters. In order to calculate user-service matrix at time t_c, we send u_i into U-LSTM and s_j into S-LSTM, for i in $[1, M]$, j in $[1, N]$. The Top-N services with the highest scores will be selected into recommendation list.

The pseudo code is shown in **Algorithm** 1.

Algorithm 1 The QF-RNN algorithm

```
Input:
    User-service invocation matrix $B_t$, user-service QoS matrix $Q1_t$, $Q2_t$, $t \in$ [1,64], user number $M$, service number $N$,the number of latent dimension $K$, and other hyperparameters.
Output:
    Recommendation list for every user
 1: for $t = 1$ to 64 do
 2:     Integrate $Q1_t$, $Q2_t$ and $B_t$ into QI-matrix $R_t$
 3:     Use matrix factorization to extract latent dimensions
 4:     Draw the User-Preference matrix $P_t$, draw the Feature-Service matrix $Q_t$
 5: end for
 6: Choose appropriate sequences as training data from $P$ and $Q$
 7: for $i = 1$ to $n_{iter}$ do
 8:     Train U-LSTM
 9:     Train S-LSTM
10:     Apply BPTT to back propagate gradient
11:     Use $\ell_2$ norm to clip gradient
12:     Update gradient
13: end for
14: for $i = 1$ to $M$ do
15:     for $j = 1$ to $N$ do
16:         Use the trained U-LSTM to calculate $\tilde{p}_{ui}$
17:         Use the trained S-LSTM to calculate $\tilde{q}_{sj}$
18:         Calculate $\tilde{r}_{us} = F(p_{ui}, q_{sj} \mid \Theta)$
19:     end for
20: end for
21: Select Top-N services with the highest score from matrix $\tilde{R}_{us} = \{\tilde{r}_{us}\}$ into recommendation list
22: return Recommandation list for every user
```

IV. Experiments

A. Dataset Description

We adopt a real-world Web service QoS dataset: WS-Dream dataset [25], which has been widely studied. It contains two types of QoS properties (response time and throughput) collected from 4,500 Web services for 142 users in 64 time intervals.

From the analysis of WS-Dream dataset as shown in Fig. 4. a, we can see that a total number of 66,617 invocations are made to the services which are invocated at one time but not invocated at the next time. Among them, 40,106 times occur in the case of throughput = 0, and 40,128 times occur in the case of respond time>1. It can be seen that the QoS value of the previous time does affect the situation of service invocation at next time.

We count the average number of services that users invoke at each time, and select some users' data to display in Fig. 4. b. It shows that a user calls up to 3,300 services averagely at a time slice. Therefore, it is more meaningful to recommend services with the highest QoS values.

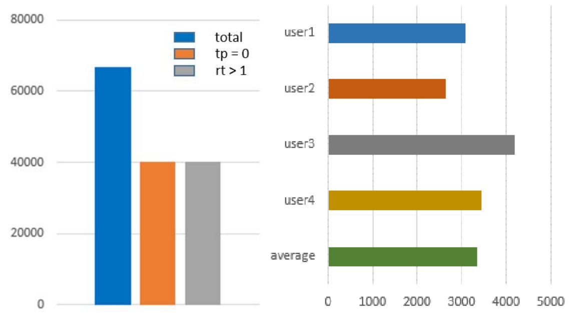

Fig. 4. The analyses of WS-Dream dataset: (a) counts the total number of services which are invocated at one time but not invocated at the next time under different constraints; (b) computes the average number of services that users invoke at each time.

B. *Evaluation Metrics*

We adopt three commonly used metrics to measure the performance of the algorithm, including precision, recall and F1. We use $\{ST_u\}$ to represent services with the highest QoS for one user. N_s is used to represent the length of $\{ST_u\}$, and $max(N_s)$ is set to 100 in our experiments. That means if a user actually invokes N services, $N_s = 100$ in the case of $N \geq 100$ and $N_s = N$ when $N < 100$.

1) Precision: Precision is defined as the ratio of high QoS services in the recommendation list to all the recommended services.

$$P = \frac{N_{rs}}{N_r} \tag{11}$$

where N_{rs} is the number of services both in recommendation list and $\{ST_u\}$, and N_r is the total number of services in the recommendation list.

2) Recall: The recall rate indicates the probability that high QoS services are recommended to the user. It is defined as the ratio of the high QoS services recommended in the list to all the services with high QoS values.

$$R = \frac{N_{rs}}{N_s} \tag{12}$$

3) F1: F1 unifies accuracy and recall, and can fully evaluate the performance of algorithms. The F1 metrics is expressed as the reconciled average of precision and recall.

$$F1 = \frac{2PR}{P+R} \tag{13}$$

where P is the precision rate and R is the recall rate.

C. *Baseline Algorithms*

In order to evaluate the performance of our QF-RNN algorithm, we compare it with four baseline methods as below.

1) **NMF**: Matrix Factorization (MF) based techniques have been widely used in Web service recommendation. Given a user-service QI-matrix $R \in R^{M\times N}$ from M users and N services, we can decompose it into two matrixes — user-preference matrix P, where $p_u \in R_K(u = 1, 2, \cdots M)$ and feature-service matrix Q, where $q_s \in R_K(s = 1, 2, \cdots N)$. Through this way, we can extract latent features at every moment. This method is not a time-aware algorithm. We can use the results of matrix factorization at last time to predict the future invocation. NMF (Non-negative Matrix Factorization) is used here [27].

2) **LDA + DQ**: Latent Dirichlet Allocation [13] is mainly used in text mining, including text topic recognition, text classification and text similarity calculation. We use users as analogy of documents, preferences as analogy of topics, and services as analogy of words, in which way LDA can be applied to service oriented computing. Firstly, we only consider the original invocation matrix when using LDA. Note that the QoS value in previous time slice will affect the user's service choice in the subsequent time slice. We use the DQ algorithm, proposed by Yanmei Zhang [20], to take the QoS value in previous time slice into consideration. Finally, we can get the score of user m on service n as Eq. 14.

$$s_{mn} = i_{mn} + (QoS_{mn})_{t-1} \tag{14}$$

where s_{mn} denotes the final score, i_{mn} is the score calculated by LDA model, and $(QoS_{mn})_{t-1}$ is the QoS observation of the last time.

3) **LSTM**: Long Short Term Memory (LSTM) is a kind of variants of the recurrent neural network (RNN), which can capture temporal dynamics from the sequence. In this comparative experiment, we directly set user-service sequence as input of LSTM. What is different in our QF-RNN model is that the input of LSTM is latent features extracted by matrix factorization.

4) **ARMA**: ARMA (Autoregressive Moving Average Model) [26] as described in Eq. 15 is a classical time series method, which will treat each service invocation as an individual. For each user-service unit in QI-matrix, we set $[r_{us,1}, r_{us,2}, \cdots, r_{us,tc-1}]$ as input. The ARMA model can thus be applied to predict $r_{us,tc}$.

$$x_t = c + \sum_{i=1}^{p} \psi_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{15}$$

where various ψ_p and θ_i are parameters of ARMA, c is a constant, and ε_t is white noise.

D. *Experimental Results Analyses*

1) Experiment 1: The purpose of this experiment is to find out how user's preferences change over time. We select four representative preferences from the experimental results to observe the changing trend, as shown in Fig. 5. Preference 1 indicates a certain degree of stability during the experimental observation period. Preference 2 shows an upward trend, while preference 3 declines in fluctuations. As to preference 4, it maintains a high value in the early stage, but hovers around a relatively low level after time slice 56. There may be several reasons for preference 4 to change in this way, such as sudden changes in throughput and response time, or other occasional leaps of environment. This experiment shows that the preferences of users fluctuate over time. Although the trend

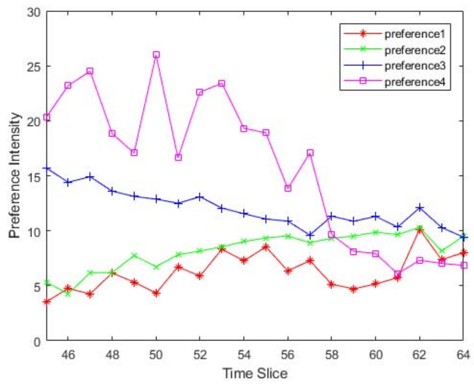

Fig. 5. Variation trend of user preferences

of the whole sequence is stable in the long run, the single value of the time series shows uncertainty in the short term. This is the deep reason why users call different services at different time slices.

2) Experiment 2: This experiment aims to compare the performance of QF-RNN with the baseline methods in terms of precision, recall and F1, respectively. Setting recommend number from 10 to 100, we can get Fig. 6, Fig. 7 and Fig. 8, respectively.

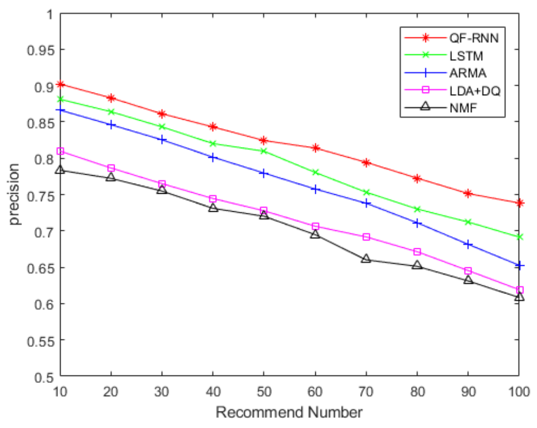

Fig. 6. Comparing of precision

Fig. 6 shows that, with the number of recommendations increasing, precision decreases. It also reveals that Our QF-RNN algorithm proposed in this paper always maintains a high accuracy rate. Besides, QF-RNN is the only method with over 90% precision. Even in the case of recommend number = 100, our method still shows a precision over 75%, which performs better than other state-of-the-art methods.

Fig. 7 shows that, as the recommend number increases, the recall rate of each algorithm increases. Although there is only a subtle difference when the number of recommendations is small, we can still observe the slight lead of our method. Moreover, with the increase of recommendations, QF-RNN

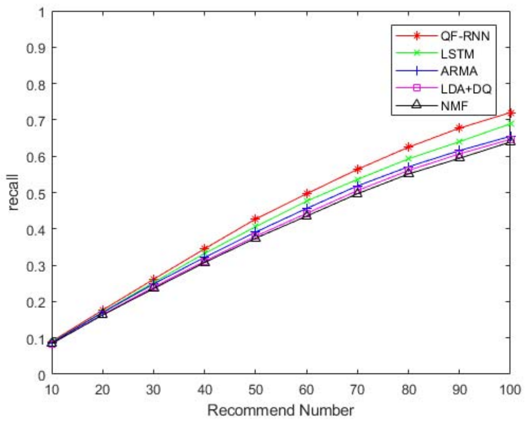

Fig. 7. Comparing of recall

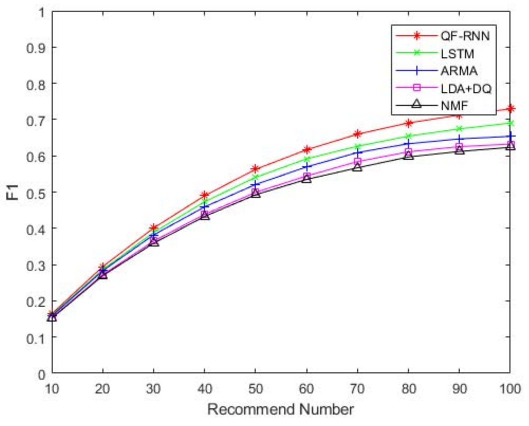

Fig. 8. Comparing of F1

algorithm has the fastest growth rate.

We can see from Fig. 8 that, the change trend of F1 is consistent with recall. Because F1 takes both precision and recall into account, the leading performance of F1 shows the powerful predicating ability of our QF-RNN algorithm.

The detailed performance of the methods tested is summarized in Table 1. We set recommend number = 50 and 100, respectively to examine the F1 indicators of various methods. When recommend number = 50, the F1 of QF-RNN, LSTM, ARMA, LDA+DQ and NMF are 56.27%, 54.06%, 52.09%, 49.26% and 49.91%, respectively. When recommend number = 100, they are 72.91%, 69.03%, 65.36%, 62.33% and 63.23%, respectively. The table shows that QF-RNN, LSTM and ARMA perform better than LDA+DQ and NMF. The reason could be that the latter are not time-aware algorithms, and they only use the invocation record and QoS observation of the last time to predict the future invocation. Among the three time-aware methods, the deep-learning powered methods perform better than ARMA. Traditional LSTM also uses deep-learning to mine long short-term dependencies of sequences.

However, when recommending services, the latent dimensions are not deeply excavated, and the changing trend of user-preferences (service-features) are not considered, instead, the end-to-end learning and prediction are carried out directly. Therefore, the overall performance of traditional LSTM is not as good as our QF-RNN. Overall, our QF-RNN method is nearly 3% better than the state-of-the art methods.

Recommendation Method	**F1@50**	**F1@100**
QF-RNN	56.27%	72.91%
LSTM	54.06%	69.03%
ARMA	52.09%	65.36%
LDA+DQ	49.26%	62.33%
NMF	49.91%	63.23%

TABLE I
COMPARING OF F1

V. CONCLUSION

In this paper, we propose a novel QI-matrix factorization-based RNN (QF-RNN) method, which uses matrix factorization to extract complicated characteristics and applies LSTM units to make time-aware service recommendation. We fully consider the influence of QoS values on user's dynamic preferences, regarding QoS as an important factor affecting recommendation results instead of only for evaluating purpose. To our best knowledge, this is the first attempt to recommend services with the highest QoS values for users. Extensive experiments have proved that our QF-RNN performs better than the baseline methods in terms of accuracy, recall and F1.

Our future work will focus on two aspects: 1) use other datasets to further validate the effectiveness of our method; 2) explore more efficient models to extract features, or use more complex neural network based on LSTM model to further improve the accuracy of recommendation.

VI. ACKNOWLEDGMENT

This research has been partially supported by the National Natural Science Foundation of China (No.61673230) and High-Tech Ship Research Project of China (17GC26102.01)

REFERENCES

[1] M.P. Papazoglou, S. Benbernou, and V. Andrikopoulos, ”On the Evolution of Services.” IEEE Transactions on Software Engineering, 38.3(2012):609-628.

[2] X. Liu and I. Fulia, ”Incorporating User, Topic, and Service Related Latent Factors into Web Service Recommendation.” Proc. IEEE Int'l Conf. Web Services (ICWS'15).

[3] M.P. Papazoglou, ”Service-Oriented Computing: Concepts, Characteristics and Directions.” Proc. Fourth Int'l Conf. Web Information Systems Engineering (WISE'03), pp.3-12,2003.

[4] W.T. Tsai, X. Zhou, Y. Chen and X.Bai, ”On Testing and Evaluating Service-Oriented Software.” Computer, 41.8(2008):40-46.

[5] X. Dong, A. Halevym, J. Madhavan, E. Nemes and J. Zhang, ”Similarity Search for Web Services.” In Proc. 13th Int'l Conf. Very Large Data Bases, Vol. 30, 2004, pp.372-383.

[6] S. Meng, W. Dou, X. Zhang, and J. Chen, ”KASR: A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Applications.” IEEE Transactions on Parallel and Distributed Systems, 25.12(2014):3221-3231.

[7] F. Gedikli and D. Jannach, ”Recommender Systems, Semantic-Based.” Springer New York, 2014.

[8] S. Chen, Y. Fan, W. Tan, J. Zhang, B. Bai and Z. Gao, ”Time-Aware Collaborative Poisson Factorization for Service Recommendation.” Proc. IEEE Int'l Conf. Web Services (ICWS'16).

[9] Y. Hao, Y. Fan, W. Tan and J. Zhang, ”Service Recommendation Based on Targeted Reconstruction of Service Descriptions.” Proc. IEEE Int'l Conf. Web Services (ICWS'17).

[10] M. Tang, Y. Jiang, J. Liu and X. Liu, ”Location-Aware Collaborative Filtering for QoS-Based Service Recommendation.” Proc. IEEE Int'l Conf. Web Services (ICWS'12).

[11] W. Tan, J. Zhang and I. Foster, ”Network Analysis of Scientific Workflows : A Gateway to Reuse.” Computer, 43.9(2010):54-61.

[12] Y. Zhou, L.Liu, C.S Perng and A. Sailer, ”Ranking Services by Service Network Structure and Service Attributes.” Proc. IEEE Int'l Conf.Web Services (ICWS'13).

[13] D.M. Blei, A.Y. Ng, and M.I. Jordan. ”Latent dirichlet allocation.” Journal of Machine Learning Research 3(2012):993-1022.

[14] Z. Gao, Y. Fan, C. Wu, W. Tan, J. Zhang, Y. Ni, B. Bai and S. Chen, ”SeCo-LDA: Mining Service Co-occurrence Topics for Composition Recommendation.” IEEE Transactions on Services Computing, 2018, PP(99):1-1.

[15] Q. Liu, E. Chen, H. Xiong, C. H. Ding and J. Chen, ”Enhancing Collaborative Filtering by User Interest Expansion via Personalized Ranking.” IEEE Transactions on Systems Man.

[16] Y. Hu, Q. Peng, X. Hu and R. Yang, ”Time Aware and Data Sparsity Tolerant Web Service Recommendation Based on Improved Collaborative Filtering.” IEEE Transactions on Services Computing, 2015, 8(5):782-794.

[17] D.M. Blei and J.D. Lafferty, ”Dynamic topic models.” Proc. Int'l Conf. Machine Learning,2006,pp.113-120.

[18] Y. Zhong, Y. Fan, K. Huang, W. Tan and J. Zhang, ”Time-Aware Service Recommendation for Mashup Creation.” IEEE Transactions on Services Computing 8.3(2015):356-368.

[19] K. Huang , Y. Fan , and W. Tan . ”Recommendation in an Evolving Service Ecosystem Based on Network Prediction.” IEEE Transactions on Automation Science and Engineering 11.3(2014):906-920.

[20] Y. Zhang, Y. Qian and Y. Wang, ”A Recommendation Algorithm Based on Dynamic User Preference and Service Quality.” Proc. IEEE Int'l Conf. Web Services (ICWS'18).

[21] S. Hochreiter and J. Schmidhuber, ”Long Short-Term Memory.” Neural computation, 9.8(1997):1735-1780.

[22] H. Lin, Y. Fan, J. Zhang and B. Bai, ”PRNN: Piecewise Recurrent Neural Networks for Predicting the Tendency of Services Invocation.” Proc. IEEE Int'l Conf. Web Services (ICWS'18).

[23] R. Xiong, J. Wang, Z. Li, B. Li and Patrick C.K. Hung,”Personalized LSTM Based Matrix Factorization for Online Qos Prediction.” Proc. IEEE Int'l Conf. Web Service (ICWS'18).

[24] R. Pascanu, T. Mikolov and Y. Bengio, ”On the difficulty of training recurrent neural networks.” Computer Science, 52.3(2012):III-1310.

[25] Z. Zheng, H. Ma, M. R. Lyu, and I. King, ”Qos-Aware Web Service Recommendation by Collaborative Filtering.” IEEE Transactions on Service Computing, 4.2(2011):140-152.

[26] G.E.P. Box and G.M. Jenkins, ”Time Series Analysis Forecating and Control.” HOlden-Day, 1970.

[27] D.D. Lee and H.S. Seung, ”Learning the Parts of Objects by Non-Negative Matrix Factorization.” Nature 401.6755(1999):788-791.

[28] Q. Yu, Z. Zheng, H. Wang, ”Trace Norm Regularized Matrix Factorization for Service Recommendation.” Proc. IEEE Int'l Conf. Web Services (ICWS'13).

[29] C. Yu, L. Huang, ”Time-Aware Collaborative Filtering for QoS-Based Service Recommendation.” Proc. IEEE Int'l Conf. Web Services (ICWS'18)..