2014 IEEE International Conference on Data Mining Workshop

An Incremental Tensor Factorization Approach for Web Service Recommendation

Wancai Zhang, Hailong Sun, Xudong Liu, Xiaohui Guo
School of Computer Science and Engineering, Beihang University
Beijing, China
Email: {zhangwc, sunhl, liuxd, guoxh}@act.buaa.edu.cn

***Abstract*—With the development of Service-Oriented technologies, the amount of Web services grows rapidly. QoS-Aware Web service recommendation can help service users to design more efficient service-oriented systems. However, existing methods assume the QoS information for service users are all known and accurate, but in real case, there are always many missing QoS values in history records, which increase the difficulty of the missing QoS value prediction. By considering the *user-service-time* three dimension context information, we study a Temporal QoS-Aware Web Service Recommendation Framework which aims to recommend best candidates to service user's requirements and meanwhile improve the QoS prediction accuracy. We propose to envision such QoS value data as a tensor which is known as multi-way array, and formalize this problem as a tensor factorization model and propose a Tucker decomposition (TD) algorithm which is able to deal with the triadic relations of *user-service-time* model. However, one major challenge is that how to deal with the dynamic incoming service QoS value data streams. Thus, we introduce the incremental tensor factorization (ITF) method which is (a) *scalable*, (b) *space efficient* (it does not need to store the past data). Extensive experiments are conducted based on our real-world QoS dataset collected on Planet-Lab, comprised of service invocation response-time values from 408 users on 5,473 Web services at 240 time periods. Comprehensive empirical studies demonstrate that our approach is faster and more accuracy than other approaches.**

I. Introduction

With the development of Service-oriented architecture (SOA), Web services have become standard software components deployed in the Internet. When developing service-oriented applications, designers try to find and reuse existing services to build the system business process. Meanwhile, multiple Web services with the same functionality are offered by different service providers. Users pick over the services based on their *Quality of Service* (QoS), such as price, availability and reputation [15].

Since selecting a high quality Web service among a large number of candidates is a non-trivial task, effective approaches to service selection and recommendation are in an urgent need. With the growing number of functionally equivalent services on the Internet, it is quite important to recommend services considering their non-functional QoS properties, such as temporal or location context.

Currently, many developers search services through public sites like Google Developers, ProgrammableWeb, Yahoo! Pipes, etc. However, none of them provide temporal QoS information for users, and such information is quite important for software deployment. Without knowledge of temporal QoS information, deployment of service-oriented software can be at great risk since the QoS of Web services may vary over the time.

However, the QoS performance of Web services observed from the user is usually quite different from that declared by the service providers due to the following reasons: 1)Web service are owned and hosted by different organizations. Furthermore, Web service invocation may produce irreversible effect in the real world. 2)There are too many service candidates, so that to evaluate all the services becomes an impossible task for service user. 3)QoS performance of Web services is highly related to user location and the invocation time, since the service status and the network environment (e.g., workload, number of clients, etc.) change over time.

In reality, service users usually only invoke a limited number of Web services and only observe the QoS value at these invocation time. So we need to predict the missing QoS values of Web services from distributed location users at different invocation time to optimize the Web services recommendation.

Based on the above analysis, we propose a Temporal QoS-Aware Web Service Prediction Framework, which employs an novel Collaborative Filtering (CF) algorithm for Web service recommendation. The CF method can predict the missing QoS value of Web services by employing their historical QoS values. But there are several challenges to be addressed when applying the temporal QoS-aware prediction: 1) How to collect the QoS information of Web services from distribution location service users? 2) How to deal with the high-dimensional QoS value data stream? 3) How to improve the CF method to adopt to our temporal QoS-aware prediction?

Contributions. We introduce tensor to Web service missing QoS value prediction: we extend the two-dimensional *user-service* matrix into a more complicated *user-service-time* triadic relations represented by three-dimensional tensor, and present a novel TF approach that is based on a generalization of MF. An Tucker decomposition algorithm and an incremental tensor factorization algorithm are proposed to predict the Web service QoS value with considering service invocation time in our Temporal QoS-Aware Web Service Recommendation Framework. Systematic evaluation on large, real-world Web service dataset, which contains more than 150 mipllions real-world Web service invocation results from 408 distributed service users on 5,473 real-world Web services at 240 time

978-1-4799-4274-9/14 $31.00 © 2014 Crown Copyright
DOI 10.1109/ICDMW.2014.176

periods. Our real-world Web service dataset has been published online[1].

II. Related Work

There is a line of work focused on the QoS optimization and Web service recommendation methods, aiming to improve the customers satisfaction. Zeng et al. first transferred the QoS-based service selection problem into an optimization problem in [15]. They used linear programming techniques to find the optimal selection of component services. Alrifai et al. combined the global optimization with local selection in order to find an approximate optimal selection in [1]. After that, they proposed a new method to select and use representative skyline services for QoS-aware Web service composition in [2].

To get personalized QoS, a few works attempted to predict the missing values with applying collaborative filtering methods. Shao et al. [12] first employed CF technique to predict the QoS of Web service. They proposed a user-based CF algorithm to predict QoS value. Zheng et al. [17] proposed a hybrid user-based and item-based CF algorithm to predict the personalized QoS, and they carried out a series of large-scale experiments based on real Web services dataset. Jiang et al. [5] proposed that the influence of personalization of Web service items should be taken into account when computing degree of similarity between users. It is that more popular services or services with more stable QoS from user to user should contribute less to user similarity measurement. Zhang et al. [7] suggested that it was better to combine users' QoS experiences, environment factor and user input factor to predict QoS value. Chen et al. [4] discovered the great influence of user's location to the accuracy of prediction and proposed a region based hybrid Collaborative Filter algorithm to predict the QoS of services. Lo et al. [8] propose an extended Matrix Factorization (EMF) framework with relational regularization to make missing QoS value prediction. They added a user-based regularization term and a service-based regularization term in the minimizing objective function.

Our work is quite different from these approaches which deal with the *user-service* two-dimensional matrix data without considering of the temporal information of Web service invocation. The absence of temporal information leads QoS prediction to suffering from the static model issue. We argue that in QoS-aware Web service recommendation, the prediction model should consider using the temporal information to reveal the triadic relations of *user-service-time*, rather than the dyadic relations of *user-service*. To learn the triadic relations from the QoS value data, a tensor factorization approach is proposed. The Tucker tensor model [14] is utilized to represent the triadic relations among user, service and time. In this paper, we deal with efficiently storing a sparse three-dimensional tensor whose elements are the QoS value according to the triplet $\langle user, service, time\rangle$ and proposing a TD method, to approximate this QoS value tensor.

III. Problem Formulation

When a service user invokes a Web service, the QoS property performance will be collected by our recommendation system. After running a period of time, the system accumulates a collection of Web service QoS value data. It can be represented by a set of quadruplets $\langle$*UserID, ServiceID, TimeID, Value*$\rangle$ (or $\langle u, s, t, v\rangle$ for short), where *UserID, ServiceID, TimeID* is defined as the three-dimensional index according to Each QoS *Value*. Using the QoS value data, a three-dimensional tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ can be constructed where I, J and K are the number of *UserID*, *ServiceID* and *TimeID*, respectively. Each entry $\boldsymbol{\mathcal{X}}_{ijk}$ in the tensor represents the QoS value of Web service j observed by the service user i at the invoking time k. $\boldsymbol{\mathcal{X}}_{ijk} = null$, if user i did not invoke Web service j previously. The missing value of Web service is a common phenomenon, even though the density of the dataset collected by our system is only 30%, so our QoS value data $\boldsymbol{\mathcal{X}}$ is a large and sparse tensor.

A. Tensor Notation

Before clarifying our model, some basic notations and operations for tensor are first introduced, which are necessary for further understanding Tucker Decomposition. tensors are denoted by bold calligraphic upper-case letters $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \cdots$, matrices by bold upper-case letters $\mathbf{A}$,$\mathbf{B} \cdots$, vectors by bold lower-case letters $\mathbf{a}$,$\mathbf{b} \cdots$ and scalars by lower-case letters $a, b \cdots$. A tensor is a multidimensional or N-way array. An N-way tensor is denoted as: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, which has N indices $(i_1, i_2, \cdots, i_N)$ and its elements are denoted by $a_{i_1 i_2\cdots i_N}$. An N-way tensor is denoted as: $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, which has N indices $(i_1, i_2, \cdots, i_N)$ and its elements are denoted by $a_{i_1 i_2\cdots i_N}$. The n-mode matricization operation maps a tensor into a matrix. The n-mode matrix of an N-way tensor $\boldsymbol{\mathcal{A}}$ are the I_n-dimensional matrix obtained from $\boldsymbol{\mathcal{A}}$ by varying the index i_n and keeping the other indices fixed, and the elements of $\boldsymbol{\mathcal{A}}$ is mapped into the unfolding matrix $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n\times(I_1 I_2\cdots I_{n-1} I_{n+1}\cdots I_N)}$.

B. Offline Tensor Factorization

We now introduce the offline tensor factorization algorithm Tucker decomposition (TD) which is a generalization of SVD for higher order tensors (HOSVD) [13].

Definition 1: **(Tucker Decomposition)** *Let $\boldsymbol{\mathcal{X}}$ be a tensor of size $I_1 \times I_2 \times \cdots \times I_N$ and $\hat{\boldsymbol{\mathcal{X}}}$ be the approximate result of tensor $\boldsymbol{\mathcal{X}}$. Tucker decomposition of $\boldsymbol{\mathcal{X}}$ yields a core tensor $\boldsymbol{\mathcal{G}}$ of specified size $J_1 \times J_2 \times \cdots \times J_N$* and factor matrices $\mathbf{A}^{(n)}$ *of size $I_n \times J_n$ for $n = 1, \cdots, N$. Thus, we have*

$$\begin{aligned}\boldsymbol{\mathcal{X}} &\approx \hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} = \boldsymbol{\mathcal{G}} \times \{\mathbf{A}\} \\ &= \sum_{i_1=1}^{J_1}\sum_{i_2=1}^{J_2}\cdots\sum_{i_N=1}^{J_N} \boldsymbol{\mathcal{G}}_{i_1 i_2\cdots i_N}\mathbf{a}^{(1)}_{i_1} \circ \mathbf{a}^{(2)}_{i_2} \circ \cdots \circ \mathbf{a}^{(N)}_{i_N}. \end{aligned} \quad (1)$$

The Tucker decomposition approximates a tensor as a smaller core tensor times the product of factor matrices that span appropriate subspaces in each mode, as illustrated in Figure 1 for $N = 3$.

Unfortunately, there is not the closed form solution for TD. A common approach for solving the problem is to use an alternating least squares (ALS) method, solving for one factor matrix at a time while holding the others fixed. In other words, we fix all the factor matrices except $\mathbf{A}^{(n)}$ and solve

$$\max_{\mathbf{A}^{(n)}} \left\| \boldsymbol{\mathcal{X}} \times \{\mathbf{A}^T\} \right\|. \quad (2)$$

[1] http://www.service4all.org.cn/servicexchange/

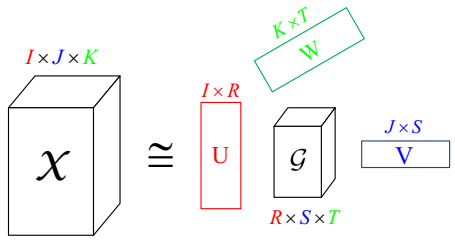

Fig. 1. Tucker decomposition of a 3-way tensor

By setting

$$\mathcal{B} = \mathcal{X} \times_{-n} \{\mathbf{A}^T\}, \tag{3}$$

we can rewritten Equation (2) as

$$\max_{\mathbf{A}^{(n)}} \left\| {\mathbf{A}^{(n)}}^T \mathbf{B}_{(n)} \right\|, \tag{4}$$

which is solved via the SVD of $\mathbf{B}_{(n)}$ and for more details see [6]. Above the derivation corresponding to ALS algorithm for Tucker decomposition, the pseudo-code of this algorithm is given by Algorithm 1.

Note that TD requires all tensors available up front, which is not possible for dynamic environments (i.e., new tensor keep coming). Even if we want to apply TD every time that a new tensor arrives, it is prohibitively expensive or merely impossible since the computation and space requirement are unbounded. Next we present an algorithm for the dynamic setting.

Algorithm 1 : ALS algorithm for Tucker decomposition

Input: the tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$,
the size vector $\mathbf{g}$ of core tensor $\mathcal{G}$ $J_1 \times J_2 \times \cdots \times J_N$.

Output: the approximate tensor $\hat{\mathcal{X}}$,
Core tensor $\mathcal{G}$,
factor matrices $\mathbf{A}^{(n)}$ of size $I_n \times J_n$.

1: **Procedure** $[\hat{\mathcal{X}}, \mathcal{G}, \mathbf{A}^{(n)}]$ = TD($\mathcal{X}, \mathbf{g}$).
2: **Initialize**: $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, for $n = 1, \cdots, N$ by small random values.
3: **Repeat**
4: **for** $n = 1, \cdots, N$ **do**
5: $\mathcal{B} = \mathcal{X} \times_{-n} \{\mathbf{A}^T\}$,
6: $\mathbf{A}^{(n)} \leftarrow J_n$ leading left singular vectors of $\mathbf{B}_{(n)}$;
7: **end for**
8: **Until** convergence or the maximum number of iterations is exceeded.
9: $\mathcal{G} = \mathcal{B} \times \{\mathbf{A}\}$.
10: $\hat{\mathcal{X}} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)}$.
11: **Return**: $\mathcal{X}, \mathcal{G}, \mathbf{A}^{(n)}$.
12: **EndProcedure**

C. Dynamic Tensor Factorization

Here we present the dynamic tensor factorization, an incremental algorithm for tensor dimensionality reduction.

Definition 2: **(Tensor Stream)** *A sequence of Mth order tensor $\mathcal{X}_1 \cdots \mathcal{X}_n$, where each $\mathcal{X}_i \in \mathbb{R}^{N_1 \times \cdots \times N_m} (1 \le i \le n)$, is called a tensor stream if n is an integer that increases with time.*

Intuitively, we can consider a tensor stream is coming incrementally over time. $\mathcal{X}_n$ is the latest tensor in the stream.

1) Incremental SVD: In the classic incremental SVD algorithm [11], the aim is to update a given SVD when new data arrives. The algorithm in [10] extends the classic incremental SVD by computing the subspace of a dynamic matrix with the mean updating which removes the assumption that the mean of the previous data is equal to the mean of the new data. In the following, this incremental SVD algorithm is briefly described. Let $\mathbf{A}'$ be the previous data matrix where the data are represented by a matrix. Let $\mathbf{F}'$ be a new data matrix. Let $\mu_{A'}, \mu_{F'}$, and μ_A be the column mean vectors of $\mathbf{A}', \mathbf{F}'$ and $(\mathbf{A}'|\mathbf{F}')$ respectively, where the operation — merges the left and the right matrices. Let $\{\mathbf{U}_{A'}, \mathbf{\Sigma}_{A'}, \mathbf{V}_{A'}\}$ be the SVD of $\mathbf{A}'$, and the SVD $\{\mathbf{U}_A, \mathbf{\Sigma}_A, \mathbf{V}_A\}$ of $(\mathbf{A}'|\mathbf{F}')$ is estimated from $\mu_{A'}, \{\mathbf{U}_{A'}, \mathbf{\Sigma}_{A'}, \mathbf{V}_{A'}\}$ and $\mathbf{F}'$. This incremental updating process is outlined as follows:

1. Compute

$$\mu_A = \frac{I_{A'}}{I_{A'} + I_{F'}} \mu_{A'} + \frac{I_{F'}}{I_{A'} + I_{F'}} \mu_{F'}$$

where $I_{A'}$ is the number of the columns in $\mathbf{A}'$ and $I_{F'}$ is the number of columns in $\mathbf{F}'$.

2. Construct a new matrix E:

$$\mathbf{E} = \left((\mathbf{F}' - \mu_{F'} l_{1 \times I_{F'}}) | \sqrt{\frac{I_{A'}}{I_{A'} + I_{F'}}} (\mu_{A'} - \mu_{F'}) \right)$$

where $l_{1 \times I_F}$ is $\overbrace{(1, 1, \cdots, 1)}^{I_{F'}}$.

3. Compute the QR decomposition of E to obtain the eigenbasis $\tilde{\mathbf{E}}$ of $\mathbf{E}$. Let matrix $\mathbf{U}'$ be $\mathbf{U}' = (\mathbf{U}_{A'}|\tilde{\mathbf{E}})$.

4. Let matrix $\mathbf{V}'$ be

$$\mathbf{V}' = \begin{pmatrix} \mathbf{V}_{A'} & 0 \\ 0 & \mathbf{\Omega}_{I_{F'}} \end{pmatrix}$$

where $\mathbf{\Omega}_{I_{F'}}$ is the $I_{F'} \times I_{F'}$ identity matrix. Then, matrix $\mathbf{\Sigma}'$ is defined as:

$$\mathbf{\Sigma}' = \begin{pmatrix} \mathbf{\Sigma}_{A'} & \mathbf{U}_{A'}^T \mathbf{E} \\ 0 & \tilde{\mathbf{E}}^T \mathbf{E} \end{pmatrix}$$

5. Compute the SVD of $\mathbf{\Sigma}'$: $\mathbf{\Sigma}' = \tilde{\mathbf{U}} \tilde{\mathbf{\Sigma}} \tilde{\mathbf{V}}^T$. Then, the SVD of $(\mathbf{A}'|\mathbf{F}')$ is obtained: $\mathbf{U}_A = \mathbf{U}' \tilde{\mathbf{U}}, \mathbf{\Sigma}_A = \tilde{\mathbf{\Sigma}}$, and $\mathbf{V}_A^T = \tilde{\mathbf{V}}^T \mathbf{V}'^T$.

2) Incremental Tensor Factorization: Based on the incremental SVD, we presented below efficiently tensor factorization algorithm which is capable of incrementally updating when new data arrive. Given a 3-order tensor $\mathcal{A}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^A}$, when a new 3-order tensor $\mathcal{F}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^F}$ arrives, $\mathcal{A}'$ is extended along the third order to form a tensor $\mathcal{A} = (\mathcal{A}'|\mathcal{F}') \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ where the operation — merges the left and the right tensors along the third order, and $I_3 = I_3^A + I_3^F$. Figure 2 illustrates the process of unfolding $\mathcal{A}$ and the relations between the previous unfolding matrices $\mathbf{A}'_{(1)}, \mathbf{A}'_{(2)}, \mathbf{A}'_{(3)}$, the new added unfolding matrices $\mathbf{F}'_{(1)}, \mathbf{F}'_{(2)}, \mathbf{F}'_{(3)}$ and the current

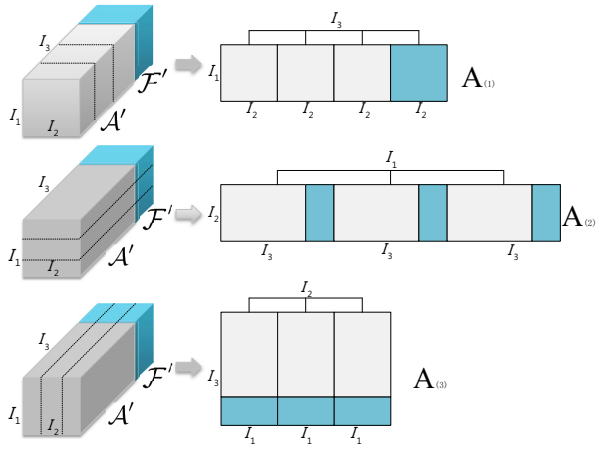

Fig. 2. Unfolding an extended 3-order tensor.

unfolding matrices $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}, \mathbf{A}_{(3)}$. The three different modes of unfolding a extended 3-order tensor are shown in the left of Figure 2. The three unfolding matrices $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}$, and $\mathbf{A}_{(3)}$ corresponding to the three different modes are shown in the right of Figure 2.

After adding of the new tensor, the column vector of the two mode-1 and mode-2 unfolding matrices are extended, and the row vector of the mode-3 matrix is extended. Our incremental tensor factorization algorithm needs to online track the changes in the three extended modes. The three modes are handled in the following ways:

1. As regards as $\mathbf{A}_{(1)}$, as $\mathbf{A}_{(1)} = (\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)})$, the SVD of $\mathbf{A}_{(1)}$ can be obtained from the SVD of $\mathbf{A}_{(1)}$ and $\mathbf{F}_{(1)}$ using the incremental SVD algorithm.

2. As regards as $\mathbf{A}_{(2)}$, it is noted that $\mathbf{A}_{(2)}$ can be decomposed as: $\mathbf{A}_{(2)} = (\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)}) \cdot \mathbf{P}$, where $\mathbf{P}$ is an orthonormal matrix obtained by column exchange and transpose operations on an identity matrix $\mathbf{Z}$ with rank $I_1 I_3$. Let

$$\mathbf{Z} = (\overbrace{\mathbf{E}_1}^{I_3^A} | \overbrace{\mathbf{Q}_1}^{I_3^F} | \overbrace{\mathbf{E}_2}^{I_3^A} | \overbrace{\mathbf{Q}_2}^{I_3^F} | \cdots | \overbrace{\mathbf{E}_{I_1}}^{I_3^A} | \overbrace{\mathbf{Q}_{I_1}}^{I_3^F})$$

which is generated by partitioning $\mathbf{Z}$ into $2I_1$ blocks along the column dimension. The partition of $\mathbf{Z}$ corresponds to $\mathbf{A}_2$ block partition shown in Figure 2 (i.e. $\mathbf{E}_1, \mathbf{E}_2, \cdots, \mathbf{E}_{I_1}$ correspond to the white regions, and $\mathbf{Q}_1, \mathbf{Q}_2, \cdots, \mathbf{Q}_{I_1}$ correspond to the gray regions. Consequently, the orthonormal matrix $\mathbf{P}$ is formulated as:

$$\mathbf{P} = (\mathbf{E}_1|\mathbf{E}_2|\cdots|\mathbf{E}_{I_1}|\mathbf{Q}_1|\mathbf{Q}_2|\cdots|\mathbf{Q}_{I_1})^T. \quad (5)$$

In this way, the $\mathbf{A}_{(2)}$ can be efficiently obtained by the SVD of $(\mathbf{A}'_{(2)}|\mathbf{F}'_{(2)})$ from the SVD of $\mathbf{A}'_{(1)}$ and $\mathbf{F}'_{(1)}$ using the incremental SVD algorithm.

3. As regards as $\mathbf{A}_{(3)}$, we estimate the row subspace, instead of the column subspace. We should calculate the SVD of the matrix $(\frac{\mathbf{A}'_{(3)}}{\mathbf{F}'_{(3)}})^T$, where - merges the upper and lower matrices. Due to $(\frac{\mathbf{A}'_{(3)}}{\mathbf{F}'_{(3)}})^T = ({\mathbf{A}'_{(3)}}^T|{\mathbf{F}'_{(3)}}^T)$, we can obtain the SVD of $\mathbf{A}_{(3)}$ from the SVD of ${\mathbf{A}'_{(3)}}^T$ and ${\mathbf{F}'_{(3)}}^T$ using the incremental SVD algorithm.

We can use the above results of $\mathbf{A}_{(1)}, \mathbf{A}_{(2)}$ and $\mathbf{A}_{(3)}$ to formulate online 3-order factorization. In summary, Algorithm 2 gives the whole factorization scheme for online tensor factorization.

Algorithm 2 : Incremental Tensor Factorization

Input: SVD of the mode-k unfolding matrix $\mathbf{A}_{(k)}$, i.e. $\mathbf{U}^{(k)}, \mathbf{\Sigma}_A^{(k)}, {\mathbf{V}^{(k)}}^T (1 \leq k \leq 3)$ of original tensor $\boldsymbol{\mathcal{A}}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^A}$ and new added tensor $\boldsymbol{\mathcal{F}}' \in \mathbb{R}^{I_1 \times I_2 \times I_3^F}$.

Output: SVD of the mode-i unfolding matrix $\mathbf{A}_{(i)}$, i.e. $\hat{\mathbf{U}}^{(i)}, \hat{\mathbf{\Sigma}}_A^{(i)}, \hat{\mathbf{V}}^{(i)T} (1 \leq i \leq 3)$ of $\boldsymbol{\mathcal{A}} = (\boldsymbol{\mathcal{A}}'|\boldsymbol{\mathcal{F}}') \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ where $I_3 = I_3^A + I_3^F$
the approximate tensor $\hat{\boldsymbol{\mathcal{A}}}$.

1: $\mathbf{A}_{(1)} = (\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)})$;
2: $\mathbf{A}_{(2)} = (\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)}) \cdot \mathbf{P}$ where $\mathbf{P}$ is defined in Eq. (5);
3: $\mathbf{A}_{(3)} = (\frac{\mathbf{A}'_{(3)}}{\mathbf{F}'_{(3)}})^T$;
4: $\hat{\mathbf{U}}^{(1)} = \mathbf{U}^{(1)}, \hat{\mathbf{\Sigma}}_A^{(1)} = \mathbf{\Sigma}_A^{(1)}$ and $\hat{\mathbf{V}}^{(1)} = \mathbf{V}^{(1)}$;
5: $\hat{\mathbf{U}}^{(2)} = \mathbf{U}^{(2)}, \hat{\mathbf{\Sigma}}_A^{(2)} = \mathbf{\Sigma}_A^{(2)}$ and $\hat{\mathbf{V}}^{(2)} = \mathbf{P}^T\mathbf{V}^{(2)}$;
6: $\hat{\mathbf{U}}^{(3)} = \mathbf{V}^{(3)}, \hat{\mathbf{\Sigma}}_A^{(3)} = {\mathbf{\Sigma}_A^{(3)}}^T$ and $\hat{\mathbf{V}}^{(3)} = \mathbf{U}^{(3)}$.

It is note that our online tensor factorization is different form the offline tensor factorization formulated in Algorithm 1. This can be regarded as an extension of the offline tensor factorization.

The first R_1, R_2 and R_3 dominant singular vectors with the larger singular values are selected from the SVD of $(\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)})$, $(\mathbf{A}'_{(2)}|\mathbf{F}'_{(2)})$ and $({\mathbf{A}'_{(3)}}^T|{\mathbf{F}'_{(3)}}^T)$ to form the approximation of $(\mathbf{A}'_{(1)}|\mathbf{F}'_{(1)})$, $(\mathbf{A}'_{(2)}|\mathbf{F}'_{(2)})$ and $({\mathbf{A}'_{(3)}}^T|{\mathbf{F}'_{(3)}}^T)$, respectively. The obtained three approximations form the result of the rank$-(R_1, R_2, R_3)$ incremental tensor factorization (ITF).

IV. Experiments

We implement the TD and ITF algorithms with Matlab. For constructing the temporal QoS value tensor and solving the Tucker decomposition, we use the Matlab Tensor Toolbox [3]. The experiments are conducted on a Dell PowerEdge T620 machine with two Intel Xeon 2.00GHz processors and 192GB RAM, running Window Server 2012.

A. Experimental Setup

To verify the prediction accuracy of our proposed approach, we implemented a prototype Web service invoke system *ServiceXChange* and a QoS evaluation middleware *QoSDetecter*. The *ServiceXChange* is a platform, more than 20,000 openly-accessible Web services obtained by crawling Web service information from Internet. Recently we have integrated it with our *Service4All*[2] platform: A Cloud Platform for Service-oriented Software Development. We employ the distributed computers from Planet-Lab to observe and collect evaluation results of the target Web services to our server. After processing the evaluation results, 408 nodes on Planet-Lab were selected as the Web service users and 5,473 publicly available real-world Web services are monitored by each node

[2]http://www.service4all.org.cn/

continuously. Finally, this experiment dataset is consist of these Web services QoS performances of 30 days from October 14 to November 21 of 2013 in 240 time intervals lasting for 3 hours.

The dataset contains more than 37 million $\langle u, s, t, v\rangle$ quadruplets, and we obtain two $408 \times 5473 \times 240$ *user-service-time* tensors which contain response time value and throughput value, respectively. Response time is defined as the persistent time between a service user sending a request and receiving the corresponding response, while throughput is defined as the average rate of successful message size per second. The statistics of Web service QoS value dataset are summarized in Table I From Table I, the means of response time and throughput is 0.6033 seconds and 8.2107 kbps, respectively.

TABLE I. STATISTICS OF WEB SERVICE QOS VALUE

Statistics	Response Time	Throughput
Scale of QoS value	0-200s	0-1000kbps
Mean of QoS value	0.6033	8.2107
Num. of service users	408	408
Num. of Web services	5473	5473
Num. of Time periods	240	240

B. Metrics

To evaluate the QoS prediction of accuracy performance, we make use of *Mean Absolute Error* (MAE) and *Root Mean Squared Error* (RMSE) [16] metrics to measure the accuracies and compare them with other commonly used CF approaches. MAE is defined as:

$$MAE = \frac{1}{N} \sum_{i,j,k} \left| \boldsymbol{\mathcal{X}}_{ijk} - \hat{\boldsymbol{\mathcal{X}}}_{ijk} \right|$$

where $\boldsymbol{\mathcal{X}}_{ijk}$ denotes actual QoS value of Web service j observed by user i at time period k, $\hat{\boldsymbol{\mathcal{X}}}_{ijk}$ represents the predicted QoS value, and N is the number of predicted elements. RMSE is defined as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i,j,k} \left(\boldsymbol{\mathcal{X}}_{ijk} - \hat{\boldsymbol{\mathcal{X}}}_{ijk} \right)^2}$$

C. Baselines

According to the prediction performance measured on the dataset, we investigate whether our method TD and ITF can be captured by the following 4 baseline algorithms:

UMean: This method uses the mean QoS value of each user to predict the missing values.

IMean: This method employs the mean QoS value of every service to predict the missing values.

WSRec: This method is a hybrid collaborative algorithm that employs both the similar users and similar services for the missing QoS value prediction [17].

RSVD: We use the regularized SVD proposed in [9].

D. Accuracy Comparison

In this part, the above 4 baseline methods are compared with TD and ITF given the same training and testing cases. Since the baselines cannot be directly applied to 3 dimensions QoS prediction problem, we employ a special formulation for making the comparison with our TD and ITF methods. We consider the *user-service-time* tensor as a set of *user-service* matrix slices in terms of time interval. Firstly, we compress the tensor into a *user-service* matrix. Each element of this matrix is the average of the specific $\langle user, service\rangle$ pair during all the time intervals. For each slice of the tensor, the baselines are applied for predicting the missing values. Secondly, we compute the MAE and RMSE of baselines, and make the comparison with TD and ITF.

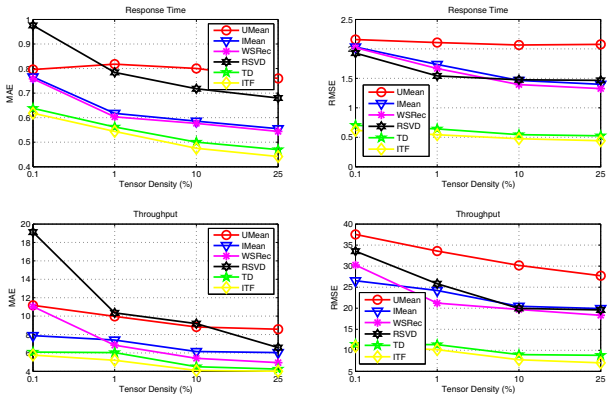

Fig. 3. Impact of Sparseness

We investigate the impact of data sparseness on the prediction accuracy as shown in Figure 3. We set the density of the training tensor as 0.1%, 1%, 10% and 25%. Figure 3 is the MAE and RMSE results of response time and throughput, and it shows that: (1) with the increase of the training density, the performance of TD and ITF enhances, indicating that better prediction is achieved with more QoS data. (2) TD and ITF outperform baselines consistently, and the reason is that these baselines only utilize the two-dimensional *user-service* model without considering the more useful triadic relations of user, service and temporal information in the *user-service-time* model. (3) The RMSE of *UMean* is much worse than that of other methods. The reason is that in our dataset the number of users is much smaller than that of services.

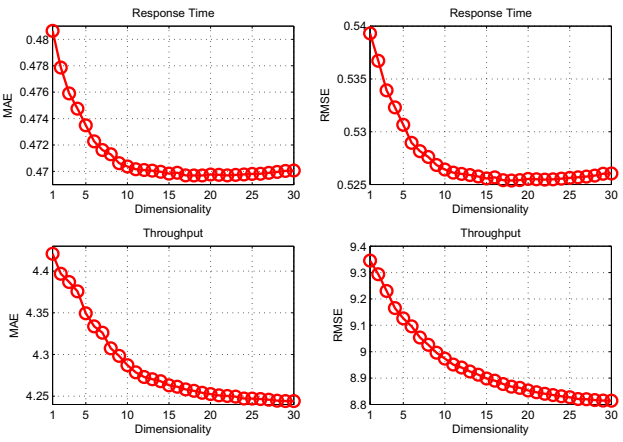

Fig. 4. Impact of Factor Matrices Dimensionality

The parameter dimensionality determines how many latent factors involve to TD. We investigate the impact of the dimensionality by setting the tensor density as 25% and varying the value of dimensionality from 1 to 30 with a step value of one. Figure 4 is the MAE and RMSE results of response time and throughput, and it shows that with the increase of factor matrix dimensionality, the MAE and RMSE keep a declining trend. These observed results coincide with the intuition that relative larger number of latent factor produce smaller error ratio. But, more factors will require longer computation time and storage space. Moreover, when the dimensionality exceeds a certain threshold, it causes the over-fitting problem, which will degrade the prediction accuracy. In this experiment, the threshold of factors is around 20 for response time.

E. Efficiency Comparison

To acquire deeper insights into TD and ITF, we present how computational time changes when setting different stream data length of the training tensor as 10, 50, 100, 240. We give a group of experiment results in Figure 5, that shows how different tensor properties affect running speed with the increase of the stream data length while setting the dimensionality of TD and ITF as 30. ITF has a good performance when TD and baselines running times keep increase with the stream data length consistently. Running time is defined as the training time of the algorithm model. Since the both methods *UMean* and *IMean* are simply compute the mean of QoS value, we only shown the comparison result of TD and ITF with both *WSRec* and *RSVD*.

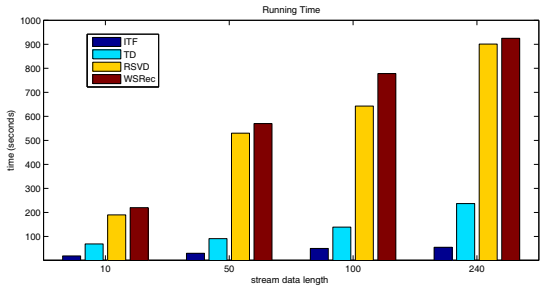

Fig. 5. Efficiency Comparison

V. Conclusions

In this paper, we employ the tensor factorization approach to advance the QoS-aware Web service prediction performance in considering of temporal information. The TD and IDF algorithms incorporate both the specially structured tensors allow for efficient storage and computation and the clustered representation of real-world QoS data. A systematic mechanism for Web service QoS value collection is designed and real-world experiments are conducted. In the experimental results, a higher accuracy of QoS value prediction is obtained with using the three-dimensional *user-service-time* model, when comparing our method with other baseline methods.

VI. Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 61103031, No. 61370057), China 863 program (No. 2012AA011203), China 973 program (No. 2014CB340300, No. 2014CB340304), A Foundation for the Author of National Excellent Doctoral Dissertation of PR China (No. 201159), and Beijing Nova Program(2011022). We are pleased to acknowledge Mingyue Liang for collecting and preparing the dataset.

References

[1] M. Alrifai and T. Risse. Combining global optimization with local selection for efficient qos-aware service composition. In *Proceedings of the 18th international conference on World wide web*, pages 881–890. ACM, 2009.

[2] M. Alrifai, D. Skoutas, and T. Risse. Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th international conference on World wide web*, pages 11–20. ACM, 2010.

[3] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. Available online, January 2012.

[4] X. Chen, X. Liu, Z. Huang, and H. Sun. Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 9–16. IEEE, 2010.

[5] Y. Jiang, J. Liu, M. Tang, and X. Liu. An effective web service recommendation method based on personalized collaborative filtering. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 211–218. IEEE, 2011.

[6] T. G. Kolda. *Multilinear operators for higher-order decompositions*. United States. Department of Energy, 2006.

[7] Z. Li, Z. Bin, L. Ying, G. Yan, and Z. Zhi-Liang. A web service qos prediction approach based on collaborative filtering. In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pages 725–731. IEEE, 2010.

[8] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu. Collaborative web service qos prediction with location-based regularization. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 464–471. IEEE, 2012.

[9] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[10] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[12] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. Personalized qos prediction forweb services via collaborative filtering. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 439–446. IEEE, 2007.

[13] J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. In *Proceedings of the 14th international conference on World Wide Web*, pages 382–390. ACM, 2005.

[14] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[15] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *Proceedings of the 12th international conference on World Wide Web*, pages 411–421. ACM, 2003.

[16] Z. Zheng, H. Ma, M. Lyu, and I. King. Collaborative web service qos prediction via neighborhood integrated matrix factorization. 6(3):289–299, July 2013.

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