2011 22nd IEEE International Symposium on Software Reliability Engineering

WSPred: A Time-Aware Personalized QoS Prediction Framework for Web Services

Yilei Zhang[†], Zibin Zheng[†] and Michael R. Lyu[†§]
[†]*Department of Computer Science and Engineering*
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{*ylzhang, zbzheng, lyu*}*@cse.cuhk.edu.hk*
[§]*School of Computer Science*
National University of Defence Technology
Changsha, China

***Abstract*—The exponential growth of Web service makes building high-quality service-oriented applications an urgent and crucial research problem. User-side QoS evaluations of Web services are critical for selecting the optimal Web service from a set of functionally equivalent service candidates. Since QoS performance of Web services is highly related to the service status and network environments which are variable against time, service invocations are required at different instances during a long time interval for making accurate Web service QoS evaluation. However, invoking a huge number of Web services from user-side for quality evaluation purpose is time-consuming, resource-consuming, and sometimes even impractical (e.g., service invocations are charged by service providers). To address this critical challenge, this paper proposes a Web service QoS prediction framework, called WSPred, to provide time-aware personalized QoS value prediction service for different service users. WSPred requires no additional invocation of Web services. Based on the past Web service usage experience from different service users, WSPred builds feature models and employs these models to make personalized QoS prediction for different users. The extensive experimental results show the effectiveness and efficiency of WSPred. Moreover, we publicly release our real-world time-aware Web service QoS dataset for future research, which makes our experiments verifiable and reproducible.**

***Keywords*-Web Service; QoS Prediction; Time-Aware**

I. Introduction

Web services are software systems designed to support interoperable machine-to-machine interaction over a network [1]. With the exponential growth of Web service as a method of communications between heterogeneous systems, Service-Oriented Architecture (SOA) is becoming a popular and major framework for building Web applications in the era of Web 2.0 [2], whereby Web services offered by different providers are discovered and integrated over the Internet. Typically, a service-oriented application consists of multiple Web services interacting with each other in several tiers. How to build high quality service-oriented applications becomes an urgent and crucial research problem.

With the growing number of Web services over the Internet, designers of service-oriented applications can choose from a broad pool of functionally identical or similar Web services when creating applications. Web services are usually deployed in remote servers and accessed by users through Internet connections. The quality of a service-oriented application, therefore, is greatly influenced by the quality of the invoked Web services. To build high-quality service-oriented applications, non-functional Quality-of-Service (QoS) performance of Web services becomes a major concern for application designers when making service selections [3]. However, the QoS performance of Web services observed from the users' perspective is usually quite different from that declared by the service providers in Service Level Agreement (SLA), due to:

- QoS performance of Web services is highly related to invocation time, since the service status (e.g., workload, number of clients, etc.) and the network environment (e.g., congestion, etc.) change over time.
- Service users are typically distributed in different geographical locations. The user-observed QoS performance of Web services is greatly influenced by the Internet connections between users and Web services. Different users may observe quite different QoS performance when invoking the same Web service.

Based on the above analysis, providing time-aware personalized QoS information of Web services is becoming more and more essential for service-oriented application designers to make service selection [3], [4], service composition [5], [6], and automatically late-binding at runtime [7].

In reality, a service user usually only invokes a limited number of Web services in the past and thus only observes QoS values of these invoked Web services. Without sufficient time-aware personalized QoS information, it is difficult for application designers to select optimal Web services at design time and replace low quality Web services with better ones at runtime. In practice, invoking Web services from users' perspectives for evaluation purpose is quite difficult, and includes the following critical drawbacks:

- Executing service invocations to obtain QoS information is too expensive for service users, since service providers may charge for invocations. At the same time,

1071-9458/11 $26.00 © 2011 IEEE
DOI 10.1109/ISSRE.2011.17

invocations for evaluation purpose consume resources of service users and service providers.
- With the growing number of Web services over the Internet, it is time-consuming to evaluate all the Web services. Moreover, some potentially appropriate Web services may not be discovered by the current user.
- To monitor the QoS performance of Web services continuously, service users need to conduct service invocations periodically, which will introduce a heavy workload to service users.
- Since service users are not experts in service evaluation, it will take a solid effort from service users to evaluate the Web services in-depth. The time-to-market constraints will also limit the amount of resources for service evaluation.

It becomes an urgent task to explore a time-aware personalized prediction approach for efficiently estimating missing QoS information of Web services for different service users. To address this critical challenge, we propose a model-based approach, called WSPred, for time-aware and personalized QoS prediction of Web services. WSPred collects time-aware QoS information from geographically distributed service users, and combines the local information to get a global user-service-time tensor. By performing tensor factorization, user-specific, service-specific and time-specific latent features are extracted from the past QoS experiences of different service users. The unknown QoS values are therefore estimated by analyzing how the user features are applied to the corresponding service features and time features. We collect a large-scale real-world Web service QoS dataset and conduct extensive experiments to compare the QoS prediction accuracy with several other state-of-the-art approaches. The experimental results show the effectiveness and efficiency of our proposed approach WSPred.

In summary, this paper makes the following contributions:

- We formally identify the critical problem of time-aware Web service QoS prediction and propose a novel collaborative framework to achieve QoS information sharing among service users. A user-side light-weight middleware is designed for automatically recording and sharing QoS experiences.
- We propose a novel time-aware personalized QoS prediction approach WSPred, which analyzes latent features of user, service and time by performing tensor factorization. We consider WSPred as the first QoS prediction approach which addresses the difference over time in service computing literature.
- We conduct large-scale real-world experiments to study the prediction accuracy and efficiency of our WSPred compared with other state-of-the-art approaches. Moreover, we publicly release our large-scale Web service QoS dataset[1] for future research. To the best of our knowledge, it is the first multi-user QoS dataset with time series information in the Web service literature.

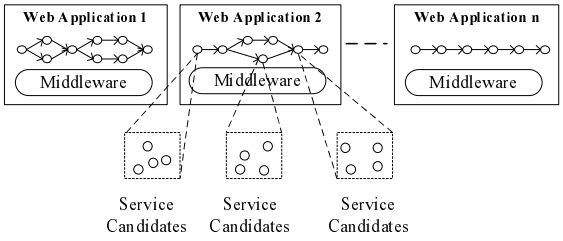


Figure 1. System Architecture

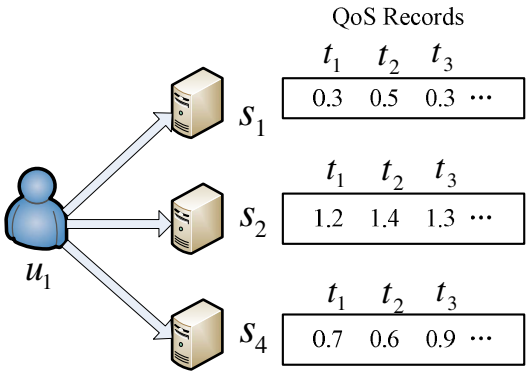


Figure 2. A Toy Example

The remainder of this paper is organized as follows: Section II describes the collaborative framework for sharing QoS information between service users. Section III presents our WSPred approach in detail. Section IV introduces the experimental results. Section V discusses related work and Section VI concludes the paper.

II. Collaborative Framework for Web Services

In this section, we present the collaborative framework for QoS prediction of Web services. Figure 1 shows the system architecture. Within a service-oriented Web application, several Web services are employed to implement complicated functions. These Web services are connected with each other in multiple tiers. For each tier, an optimal Web service will be selected from a set of functional equivalent service candidates. Typically the Web service candidates are provided by different organizations and are distributed in different geographic locations and time zones. When invoked through communication links, the user-side usage experiences are influenced by the network environments and the server-side status at invocation time.

The mechanism proposed in this paper is to (1) share local Web service usage experiences from different service users, (2) combine these pieces of local information together to get

[1] http://www.wsdream.net/

global QoS information for all service candidates, (3) extract time-specific user features and service features, and (4) make personalized time-aware QoS value prediction based on these features. As shown in Figure 2, each service user keeps local records of QoS usage experience on Web services and is encouraged to contribute its local records to obtain records from other users. By contributing more individually observed Web service QoS information, a service user can obtain more global QoS information from other users, thus obtaining more accurate Web service QoS prediction values. Given accurate QoS prediction results, service users could select the potentially optimal services for composing service-oriented Web applications. The detailed collaborative prediction approach will be presented in Section III.

Since most of the service users are not experts in service testing, to reduce the efforts of service users spent on testing the service QoS performance, we design a user-side light-weight middleware for service users to automatically record QoS values of invocations and to contribute the local records to the server for obtaining more invocation results from other service users. Within the middleware, there are three management components: *Monitor*, *Collector* and *Predictor*. *Monitor* is responsible for monitoring the QoS performance of Web services when users sends invocations. *Collector* is responsible for contributing local QoS information to other users and for collecting shared QoS information from other users. *Predictor* is responsible for providing time-aware personalized QoS value prediction based on local and other users' QoS information collected by *Collector*.

III. Time-Aware QoS Prediction

Previous Web service related techniques such as selection, composition, and orchestration only employ average QoS performance of service candidates at design-time. In recent Web service literatures, most of the state-of-the-art techniques can automatically update corresponding Web services with better ones at run-time, which requires time-specific QoS performance of Web services.

In this section, we first formally describe the QoS value prediction problem on Web services in Section III-A. Then we propose a latent feature learning algorithm to learn the user-specific, service-specific, and time-specific features in Section III-B. The missing QoS values are predicted by applying the proposed algorithm WSPred in Section III-C. Finally, the complexity analysis is conducted in Section III-D.

A. Problem Description

Figure 2 illustrates a toy example of the QoS prediction problem we study in this paper. In this figure, user u_1 has used three Web services s_1, s_2 and s_4 in the past. u_1 recorded the observed QoS performance of Web services s_1, s_2 and s_4 with specific invocation time in local site. By integrating all the QoS information from other users, we form a three-dimensional user-service-time tensor as shown in Figure 3. In this example, totally there are 5 users (from u_1 to u_5), 5 services (from s_1 to s_5) and 5 time intervals (from t_1 to t_5). The tensor is split into several slices with each one representing a time interval. Within a slice, each entry denotes an observed QoS value of a Web service from a user during the specific time interval. The problem we study in this paper is how to efficiently and precisely predict the missing entries in the user-service-time tensor based on the existing entries.

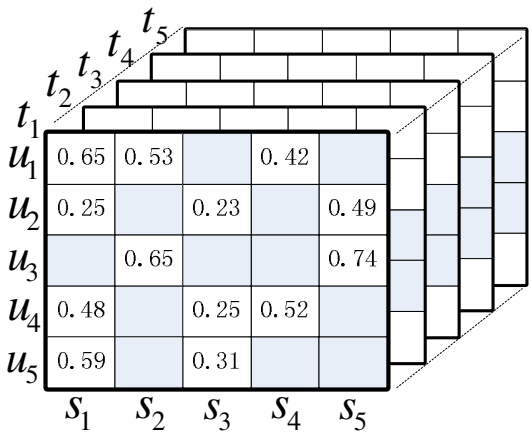


Figure 3. User-Service-Time Tensor

Now we formally define the problem of QoS prediction for Web services as follows: Given a set of users and a set of Web services, based on the existing QoS values from different users, predict the missing QoS values of Web services when invoked by users at different time intervals. More precisely:

> *Let U be the set of m users, S be the set of n Web services, and T be the set of c time intervals. A QoS element is a quartet (i, j, k, q_{ijk}) representing the observed quality of Web service s_j by user u_i at time interval t_k, where $i \in \{1, \cdots, m\}$, $j \in \{1, \cdots, n\}$, $k \in \{1, \cdots, c\}$ and $q_{ijk} \in \mathbb{R}^p$ is a p-dimensional vector representing the QoS values of p criteria. Let Ω be the set of all triads $\{i, j, k\}$ and Λ be the set of all known triads (i, j, k) in Ω. Consider a tensor $Y \in \mathbb{R}^{m \times n \times c}$ with each entry Y_{ijk} representing the observed p^{th} criterion value of service s_j by user u_i at time interval t_k. Then the missing entries $\{Y_{ijk} | (i, j, k) \in \Omega - \Lambda\}$ should be predicted based on the existing entries $\{Y_{ijk} | (i, j, k) \in \Lambda\}$.*

Typically, the QoS values can be integers from a given range (e.g., $\{0, 1, 2, 3\}$) or real numbers. Without loss of generality, we can map the QoS values to the interval $[0, 1]$

using the following function:

$$f(x) = \begin{cases} 0, & \text{if } x < Y_{min} \\ 1, & \text{if } x > Y_{max} \\ \frac{x - Y_{min}}{Y_{max} - Y_{min}}, & \text{otherwise} \end{cases}$$

where Y_{max} and Y_{min} are the specified upper bound and lower bound of QoS values respectively.

B. Latent Features Learning

In order to learn the latent features of users, services, and time, we employ tensor factorization technique to fit a factor model to the user-service-time tensor. The factorized user-specific, service-specific and time-specific matrices are utilized to make further missing entries prediction. The idea behind the factor model is to derive a high-quality low-dimensional feature representation of users, services and time by analyzing the user-service-time tensor. The premise behind a low-dimensional factor model is that there is only a small number of factors influencing QoS usage experiences, and that a user's QoS usage experience vector is determined by how each factor applies to that user, the corresponding service and the specific time interval. Examples of physical feature are network distance between the user and the server, the workload of the server, etc. Latent features are orthogonal representing the decomposed results of physical factors.

In the paper, we consider an $m \times n \times c$ QoS tensor consisting of m users, n services and c time intervals. A low-rank tensor factorization approach seeks to approximate the QoS tensor Y by a multiplication of l-rank factors [8],

$$Y \approx C \times_u U \times_s S \times_t T, \tag{1}$$

where $C \in \mathbb{R}^{l \times l \times l}$, $U \in \mathbb{R}^{m \times l}$, $S \in \mathbb{R}^{n \times l}$ and $T \in \mathbb{R}^{c \times l}$ are latent feature matrices. l is the number of latent features. Each column in U, S and T representing a user, a Web service and a time interval, respectively. $\times_u$, $\times_s$ and $\times_t$ are tensor-matrix multiplication operators with the subscript showing in which direction on the tensor to multiply the matrix (i.e., $C \times_u U = \sum_{i=1}^{l} C_{ijk} U_{ij}$). C is set to the diagonal tensor:

$$C = \begin{cases} 1, & \text{if } i = j = k \\ 0, & \text{otherwise} \end{cases}$$

Typically, $l \ll mnc$ since in the real world, each user has invoked only a small portion of Web services, and the tensor Y is usually very sparse. From the above definition, we can see that the low-dimensional matrices U, S and T are unkonwn and need to be estimated.

To estimate the quality of tensor approximation, we need to construct a loss function for evaluating the error between the estimated tensor and the original tensor. The distance between two tensors is usually employed to define the loss function:

$$\frac{1}{2} ||Y - \hat{Y}||_F^2, \tag{2}$$

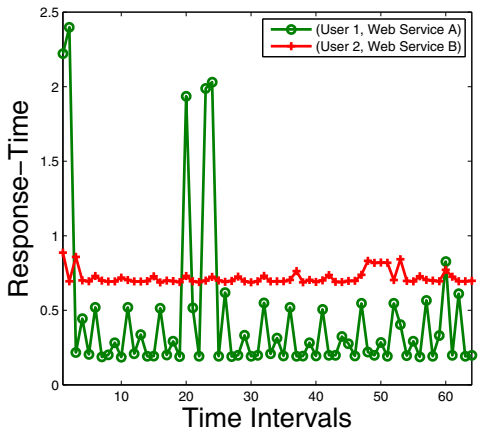


Figure 4. Response-Time of Two Pairs of User-Service

where $|| \cdot ||_F^2$ denotes the Frobenius norm. However, due to the reason that there are a large number of missing values, we only factorize the observed entries in tensor Y. Hence, we employ the following loss function instead:

$$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{c} I_{ijk} (Y_{ijk} - \hat{Y}_{ijk})^2, \tag{3}$$

where I_{ijk} is the indicator function that is equal to 1 if user u_i invoked service s_j during the time interval t_k and equal to 0 otherwise. To avoid the overfitting problem, we add three regularization terms to Eq. (3) to constrain the norms of U, S and T. Hence we conduct the tensor factorization as to solve the following optimization problem:

$$\begin{aligned} \min_{U,S,T} \mathcal{L}(Y, U, S, T) &= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{c} I_{ijk} (Y_{ijk} - \hat{Y}_{ijk})^2 \\ &+ \frac{\lambda_1}{2} ||U||_F^2 + \frac{\lambda_2}{2} ||S||_F^2 + \frac{\lambda_3}{2} ||T||_F^2, \end{aligned} \tag{4}$$

where $\lambda_1, \lambda_2, \lambda_3 > 0$. The optimization problem in Eq. (4) minimizes the sum-of-squared-errors objective function with quadratic regularization terms.

Figure 4 gives a comprehensive illustration of the Web service response-time observed by different service users. We randomly select two service users (User 1 and User 2) and two real-world Web services (Web Service A and Web Service B) from the experiment described in Section IV. As shown in Figure 4, during different time intervals, a user has different QoS experiences on the same Web service. In general, the differences are limited within a range (e.g., most of the response-time values of (User 1, Web Service A) are within the range of 0.2-0.6s and most of the response-time values of (User 2, Web Service B) are within the range of 0.7-0.9s). This observation indicates that although the QoS values of a particular user-service are different during

different time intervals, they fluctuate around the average QoS value of the user-service pair during all time intervals. Based on this observation, we further add a regularization term to Eq. (4) to prevent the predicted QoS values from varying a lot against the average QoS value. We define the prediction with average QoS value constraint as the following optimization problem:

$$\begin{aligned}\min_{U,S,T} \mathcal{L}_{\mathcal{A}}(Y,U,S,T) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{c} I_{ijk}(Y_{ijk}-\hat{Y}_{ijk})^2 \\ &+ \frac{\lambda_1}{2}||U||_F^2 + \frac{\lambda_2}{2}||S||_F^2 + \frac{\lambda_3}{2}||T||_F^2 \\ &+ \frac{\eta}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{c} I_{ijk}(\hat{Y}_{ijk}-\bar{Y}_{ij})^2, \end{aligned} \tag{5}$$

where $\eta > 0$, and $\bar{Y}_{ij}$ denotes the average QoS value of Web service s_j observed by user u_i during all the time intervals. η controls how much the prediction method should engage the information of average QoS performance. In the extreme case, if we use a very small value of η, we only perform tensor factorization without considering the global QoS information. On the other side, if we use a very large value of η, the average QoS performance will dominate the learning processes.

A local minimum of the objective function given by Eq. (5) can be found by performing incremental gradient descent in feature vectors U_i, S_j and T_k:

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial U_{if}} &= \sum_{j=1}^{n}\sum_{k=1}^{c} I_{ijk}(\hat{Y}_{ijk}-Y_{ijk})S_j^T T_k + \lambda_1 U_{if} \\ &+ \eta\sum_{j=1}^{n}\sum_{k=1}^{c} I_{ijk}(\hat{Y}_{ijk}-\bar{Y}_{ij})S_j^T T_k, \\ \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial S_{jf}} &= \sum_{i=1}^{m}\sum_{k=1}^{c} I_{ijk}(\hat{Y}_{ijk}-Y_{ijk})U_i^T T_k + \lambda_2 S_{jf} \\ &+ \eta\sum_{i=1}^{m}\sum_{k=1}^{c} I_{ijk}(\hat{Y}_{ijk}-\bar{Y}_{ij})U_i^T T_k, \\ \frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial T_{kf}} &= \sum_{i=1}^{m}\sum_{j=1}^{n} I_{ijk}(\hat{Y}_{ijk}-Y_{ijk})U_i^T S_j + \lambda_3 T_{kf} \\ &+ \eta\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ijk}(\hat{Y}_{ijk}-\bar{Y}_{ij})U_i^T S_j. \end{aligned} \tag{6}$$

Algorithm 1 shows the iterative process for latent feature learning. We first initialize matrices U, S and T with small random non-negative values. Iteration of the update rules derived from Eq. (6) converges to a local minimum of the objective function given in Eq. (5).

Algorithm 1: Latent Features Learning Algorithm

Input: Y, l, λ, η
Output: U, S, T

Initialize $U \in \mathbb{R}^{l\times m}$, $S \in \mathbb{R}^{l\times n}$ and $T \in \mathbb{R}^{l\times c}$ with small random numbers;
repeat
  for *all* $(i,j,k) \in \Lambda$ **do**
    $\hat{Y}_{ijk} = \sum_{f=1}^{l} U_{if}S_{jf}T_{kf}$;
  end
  for *all* (i,j) **do**
    $\bar{Y}_{ij} = \frac{\sum_{k=1}^{c} I_{ijk}Y_{ijk}}{\sum_{k=1}^{c} I_{ijk}}$;
  end
  for *all* $(i,j,k) \in \Lambda$ **do**
    for $(f = 1; f \le l; f++)$ **do**
      $U_{if} \leftarrow U_{if} - [(\hat{Y}_{ijk} - Y_{ijk})S_j^T T_k + \lambda U_{if} + \eta(\hat{Y}_{ijk} - \bar{Y}_{ij})S_j^T T_k]$;
      $S_{jf} \leftarrow S_{jf} - [(\hat{Y}_{ijk} - Y_{ijk})U_i^T T_k + \lambda S_{jf} + \eta(\hat{Y}_{ijk} - \bar{Y}_{ij})U_i^T T_k]$;
      $T_{kf} \leftarrow T_{kf} - [(\hat{Y}_{ijk} - Y_{ijk})U_i^T S_j + \lambda T_{kf} + \eta(\hat{Y}_{ijk} - \bar{Y}_{ij})U_i^T S_j]$;
    end
  end
until *Converge* ;

C. Missing Value Prediction

After the user-specific, service-specific and time-specific latent feature spaces U, S and T are learned, we can predict the QoS performance of a given service observed by a user during the specific time interval. For the missing entry Y_{ijk} in the QoS tensor, the value predicted by our method is defined as

$$\hat{Y}_{ijk} = I_{ijk}\sum_{f=1}^{l} U_{if}S_{jf}T_{kf}. \tag{7}$$

D. Complexity Analysis

The main computation of gradient methods is evaluating the objective function $\mathcal{L}_{\mathcal{A}}$ and their gradients against variables. The computational complexity of evaluating the objective function $\mathcal{L}_{\mathcal{A}}$ is $O(\rho_Y l + \rho_Y c)$, where ρ_Y is the number of nonzero entries in the tensor Y, l is the dimensions of the latent features, and c is the number of time intervals. The computational complexities for the gradients $\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial U}$, $\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial S}$ and $\frac{\partial \mathcal{L}_{\mathcal{A}}}{\partial T}$ in Eq. (6) are $O(\rho_Y l + \rho_Y c)$. Therefore, the total computational complexity in one iteration is $O(\rho_Y l + \rho_Y c)$, which indicates that theoretically, the computational time of our method is linear with recept to the number of observed QoS values in the user-service-time tensor Y. Note that because of the sparsity of Y, $\rho_Y << mnc$, which indicates that the computation time grows slowly with respect to the size of Tensor Y. This complexity analysis shows that our proposed approach is very efficient and can be applied to large-scale systems.

IV. EXPERIMENTS

In this section, we conduct several experiments to compare our approach with several state-of-the-art collaborative filtering prediction methods. In the following, Section IV-A introduces the experimental setup and gives the description of our experimental dataset, Section IV-B defines the evaluation metrics, Section IV-C compares the prediction quality of our approach with other competing methods, and Section IV-D and Section IV-E study the impact of tensor density and dimensionality, respectively.

A. Experimental Setup and Dataset Collection

To evaluate our proposed QoS prediction approach in the real-world, we implement a tool WSMonitor for monitoring the QoS performance of Web service, and collect a large-scale Web service QoS dataset for conducting various experiments.

WSMonitor is implemented and deployed with JDK 6.0, Eclipse 3.3, Axis 2, and Apache 2.2.17. WSMonitor first crawls a set of WSDL files from the Internet and generates a list of openly-accessible Web services. For each Web service in the list, WSMonitor automatically generates a java class for service invocation by employing the WSDL2Java tool from the Axis package [9]. Totally, 5,871 classes are generated for 5,871 Web services. By calling the functions within a class, null operation requests are sent to the corresponding Web service for capturing the QoS performance.

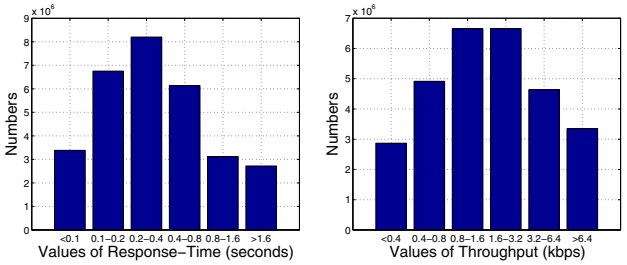
We deploy the WSMonitor on 142 distributed computers located in 22 countries from PlanetLab[2], which is a distributed test-bed consisting of hundreds of computers all over the world. Totally, 4,532 publicly available real-world Web services from 57 countries are monitored by each computer continuously. 1,339 of the initially selected Web services are excluded in this experiment due to: 1) authentication required and 2) permanent invocation failure (e.g., the Web service is shutdown). In our experiment, each of the 142 computers sends null operation requests to all the 4,532 Web services during every time interval. The experiment lasts for 16 hours with a time interval lasting for 15 minutes.

By collecting invocation records from all the computers, finally we include 30,287,611 QoS performance results into the Web service QoS dataset. Each invocation record is a k dimension vector representing the QoS values of k criteria. We then extract a set of $142 \times 4532 \times 64$ user-service-time tensors, each of which stands for a particular QoS property, from the QoS invocation records. For simplicity, we employ two tensors, which represent response-time and throughput QoS criteria respectively, for experimental evaluation in this paper. Without loss of generality, our approach can be easily extended to include more QoS criteria.

[2]http://www.planet-lab.org

Table I
STATISTICS OF WS QOS DATASET

Statistics	Response-Time	Throughput
Scale	0-20s	0-1000kbps
Mean	3.165s	9.609kbps
Num. of Users	142	142
Num. of Web Services	4,532	4,532
Num. of Time Intervals	64	64
Num. of Records	30,287,611	30,287,611



(a) (b)

Figure 5. QoS Value Distributions

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

B. Metrics

We assess the prediction quality of our proposed approach in comparison with other methods by computing Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The metric MAE is defined as:

$$MAE = \frac{\sum_{ijk} |\hat{Y}_{ijk} - Y_{ijk}|}{N}, \tag{8}$$

and RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{ijk} (\hat{Y}_{ijk} - Y_{ijk})^2}{N}}, \tag{9}$$

where Y_{ijk} is the QoS value of Web service s_j observed by user u_i at time interval t, $\hat{Y}_{ijk}$ denotes the QoS value of Web service s_j would be observed by user u_i at time interval t_k as predicted by a method, and N is the number of predicted QoS values.

C. Performance Comparisons

In this section, in order to show the effectiveness of our proposed Web service QoS prediction approach, we compare the prediction accuracy of the following methods:

1) **MF1**-This method considers the user-service-time tensor as a set of user-service matrix slices in terms of time. For each slice, the prediction method proposed by Lee and Seuing in [10] is employed. It applies non-negative matrix factorization on user-item matrix for missing value prediction.
2) **MF2**-This method first compresses the user-service-time tensor into a user-service matrix. For each entry in the matrix, the value is the average of the specific user-service pair during all the time intervals. After obtaining the compressed user-service matrix, it applies the non-negative matrix factorization technique proposed by Lee and Seuing [10] on user-item matrix for missing value prediction.
3) **TF**-This is a tensor factorization-based prediction method. It applies tensor factorization on user-service-time tensor to extract user-specific, service-specific and time-specific characterizes. The missing value is then predicted based on how these characterized apply to each other.
4) **WSPred**-This method is proposed in this paper. It is a tensor factorization-based recommendation with average QoS value constraints.

Since memory-based approaches require much more computation time than model-based approaches, we only compare the above four model-based approaches. Since the matrix factorization technique cannot be directly applied to time-aware prediction problem, we extend the prediction approach [10] in two different ways, which derive method MF1 and MF2 respectively.

In order to evaluate the performance of different approaches in reality, we randomly remove some entries from the tensors and compare the values predicted by a method with the original ones. The tensors with missing values are in different densities. For example, 10% means that we randomly remove 90% entries from the original tensor and use the remaining 10% entries to predict the removed entries. The prediction accuracy is evaluated using Eq. (8) and Eq. (9) by comparing the original value and the predicted value of each removed entry. The values of λ and η are tuned by performing cross-validation [11] on the observed QoS data. Without lost of generality, the parameter settings of all the approaches are $l = 20$ and $\lambda 1 = \lambda 2 = \lambda 3 = \eta = 0.001$ in the experiments conducted in this paper. Detailed impact of tensor density and dimensionality is studied in Section IV-D and Section IV-E.

The QoS value prediction accuracies evaluated by MAE and RMSE are shown in Table II. For each row in the table, we highlight the best performer among all methods. From Table II, we can observe that the tensor factorization-based prediction methods (i.e., TF and WSPred) outperform the matrix factorization-based prediction methods (i.e., MF1 and MF2), since the tensor factorization-based methods use the time-specific features as additional information. We also observe that our approach WSPred constantly performs better (smaller MAE and RMSE values) than the other approaches, including TF, for both response-time and throughput under both dense tensors and sparse tensors. This demonstrates the advantage of time-aware prediction algorithm with the constraints of average QoS performance. In Table II, the MAE and RMSE values of dense tensors (e.g., tensor density is 45% or 50%) are smaller than those of sparse tensors (e.g., tensor density is 5% or 10%), since a denser tensor provides more information for predicting the missing values. In general, the MAE and RMSE values of throughput are larger than those of response-time because the scale of throughput is 0-1000 kbps, while the scale of response-time is 0-20 seconds. Compared with other methods, the improvements of our approach WSPred are significant, which demonstrates that the idea of considering time information for QoS prediction is realistic and reasonable.

D. Impact of Tensor Density

In Figure 6, we compare the prediction accuracy of all the methods under different tensor densities. We change the tensor density from 5% to 50% with a step value of 5%. The parameter settings in this experiment are $l = 20$ and $\lambda 1 = \lambda 2 = \lambda 3 = \eta = 0.001$.

Figure 6(a) and Figure 6(b) show the experimental results of response-time, while Figure 6(c) and Figure 6(d) show the experimental results of throughput. The experimental results show that our approach WSPred achieves higher prediction accuracy (lower MAE and RMSE values) than other competing methods under different tensor density. In general, when the tensor density is increased from 5% to 20%, the prediction accuracy of our approach WSPred is significantly enhanced. When the tensor density is further increased from 20% to 50%, the enhancement of prediction accuracy is quite limited. This observation indicates that when the tensor is very sparse, collecting more QoS information will greatly enhance the prediction accuracy, which further demonstrates that considering both the difference between time intervals and the average QoS performance could effectively provide personalized QoS estimation.

In the experimental results, we observe that the performance of MF1 is worse than that of other methods. The reason is that MF1 only extracts the user-specific and service-specific features without considering the relationship between QoS performance in time intervals. In general, MF2 performs better than MF1, since MF2 computes the average QoS performance before performing matrix factorization. Applying the features extracted from the original tensor, MF2 predicts the average QoS performance for a particular user-service pair. This observation further demonstrates that the average QoS performance of a particular user-service pair can provide valuable information when predicting the missing QoS value of the user-service pair in a particular time interval.

Table II
PERFORMANCE COMPARSIONS (A SMALLER MAE OR RMSE VALUE MEANS A BETTER PERFORMANCE)

Tensor Density	Metrics	Response-Time (seconds)				Throughput (kbps)			
		MF1	MF2	TF	WSPred	MF1	MF2	TF	WSPred
5%	MAE	3.4137	2.9187	2.9184	**2.5580**	10.5460	8.8317	8.7997	**8.2761**
	RMSE	5.3423	5.1024	4.7508	**4.3626**	46.6735	43.4769	39.5133	**39.0962**
10%	MAE	2.8518	2.8421	2.7888	**2.4990**	9.9839	8.7522	8.5080	**8.0131**
	RMSE	5.0667	4.5563	4.5696	**4.2892**	46.6656	39.7740	39.2792	**38.6251**
45%	MAE	2.4241	2.2679	2.2511	**2.1462**	8.6773	7.9590	7.9471	**6.9398**
	RMSE	4.3240	4.2541	4.2071	**3.9200**	45.0077	39.9388	38.6964	**36.5724**
50%	MAE	2.3959	2.2596	2.2127	**2.1266**	8.6224	7.8306	7.8045	**6.8558**
	RMSE	4.2996	4.1490	4.0169	**3.8943**	44.9407	38.9388	38.6964	**36.5724**

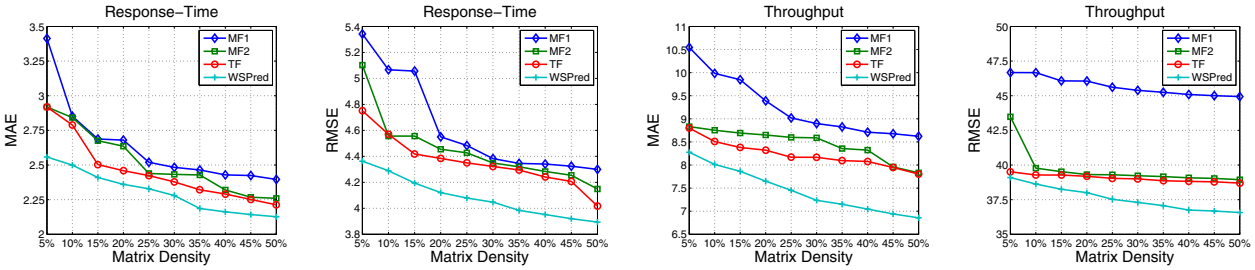


Figure 6. Impact of Tensor Density

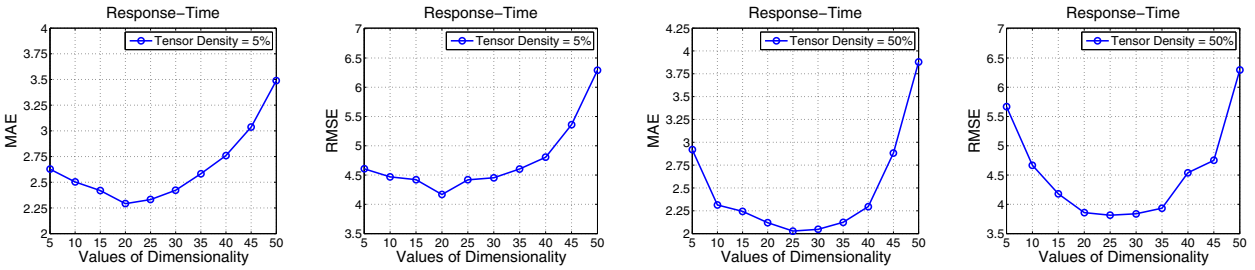


Figure 7. Impact of Dimensionality in Response-Time Dataset

E. Impact of Dimensionality

The parameter dimensionality l determines the number of latent features applied to characterize user, service and time. In Figure 7 and Figure 8, we study the impact of parameter dimensionality by varying the values of l from 5 to 50 with a step value of 5. Other parameter settings are $\lambda 1 = \lambda 2 = \lambda 3 = \eta = 0.001$.

Figure 7 and Figure 8 show the MAE and RMSE values of response-time and throughput respectively. We observe that in both figures, as l increases, the MAE and RMSE decrease (prediction accuracy increases), but when l surpasses a certain threshold like 20, the MAE and RMSE increase (prediction accuracy decreases) with further increase of the value of l. This observation indicates that too few latent factors are not enough to characterize the features of user, service and time, while too many latent factors will cause an overfitting problem. There exists an optimal value of l for characterizing the latent features. In both Figure 7 and Figure 8, when the tensor density is 50%, we observe that our approach WSPred achieves the best performance when the value of dimensionality is 25, while smaller values like 5 or larger values like 50 can potentially reduce the prediction accuracy. When the tensor density is 5%, we observe that the prediction accuracy of our approach WSPred achieves the best performance when the value of dimensionality is 20, while smaller values like 5 or larger values like 50 can potentially reduce the prediction accuracy. This observation indicates that when the user-service-time tensor is sparse, 20 latent factors are already enough to characterize the features of user, service and time which are mined from the limited user-service-time QoS information. On the other hand, when the tensor is dense, more latent factors, like 25, are needed to characterize the latent features since more QoS information

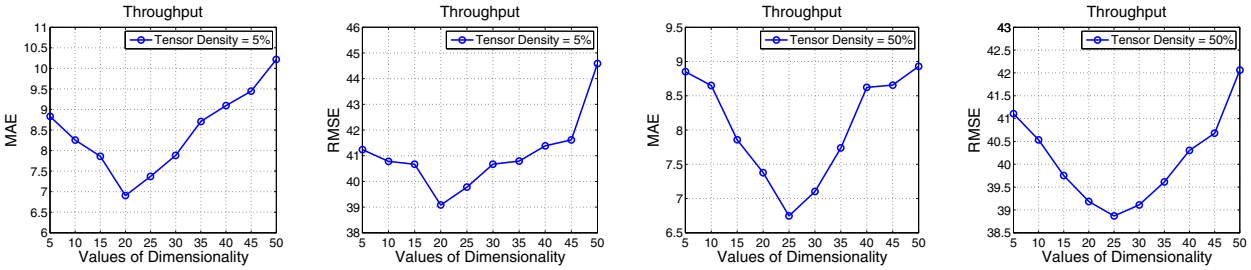


Figure 8. Impact of Dimensionality in Throughput Dataset

can be obtained from the original tensor.

V. Related Work

Web application has been in spotlight recently. SOA has become a popular framework for building Web applications. A number of research investigations are focusing on different kinds of issues such as Web service selection [3], [4], Web service composition [5], [6], failure prediction [12], reliability prediction [13], etc. Traditionally, reliability of a system [14], [15] is analyzed without considering the QoS performance, which is not accurate when applied to modern systems. Recently, Quality-of-Service (QoS) has been widely employed for representing the nonfunctional characteristics of Web services [4], [16]. QoS performance of Web services can be measured either from the service provider's perspective or from the user-side. QoS properties (e.g. price, availability, etc.) measured at the service provider side are usually identical for different users, while QoS properties (e.g., response-time, throughput, failure probability, etc.) observed by different users may vary significantly due to the unpredictable communication links and heterogeneous network environments. Based on the QoS performance of Web services, several approaches have been proposed for Web service selection [3], [4] which enable service users to identify optimal Web service from a set of functionally similar or identical Web service candidates for improving the whole quality of Web applications.

The above approaches usually assume that the values of user-dependent QoS properties are already known. To obtain the QoS values, user-side Web service evaluations are required [17]. However, in reality a user typically has engaged a limited number of Web services in the past and cannot exhaustively invoke all the available Web service candidates. In this paper, we focus on predicting missing QoS values by collaborative filtering approaches to enable the optimal Web service selection.

Collaborative filtering approaches are widely adopted in commercial recommender systems [18], [19]. Generally, traditional recommendation approaches can be categorized into two classes: memory-based and model-based. Memory-based approaches, also known as neighborhood-based approaches, are one of the most popular prediction methods in collaborative filtering systems. Memory-based methods employ similarity computation based on past usage experiences to find similar users and services for making the QoS value prediction. The most analyzed examples of memory-based collaborative filtering include user-based approaches [20], [21], item-based approaches [22], [23], and their fusion [13], [24], [25]. In [13], the reliability of active user is predicted based on the reliability of similar users found. However, the method proposed in [13] only considers two dimensions (i.e., user and Web service) while time factor is not included. Moreover, the high computational complexity makes it difficult to extend memory-based approaches (e.g. approach proposed in [13]) to handle large amounts of time-aware QoS data.

Model-based approaches employ machine learning techniques to fit a predefined model based on the training datasets. Model-based approaches include several types: the clustering models [26], the latent factor models [27], the aspect models [28], etc. Lee et al. [10] presented an algorithm for non-negative matrix factorization that is able to learn the parts of facial images and semantic features of text. The premise behind a low-dimensional factor model is that there is only a small number of factors influencing the QoS values in the user-service-time tensor, and that a user's factor vector is determined by how much each factor applies to that user. For three-dimensional tensor data, tensor factorization techniques are employed for item recommendation [8].

The memory-based approaches employ the information from similar users and services for predicting missing values. When the number of users or services is too small, similarity computation for finding similar users or services is not accurate. When the number of users or services is too large, calculating similarity values for each pair of users or services is time-consuming. In contrast, model-based approaches are very efficient for missing value prediction, since they assume that only a small number of factors influence the QoS performance. In this paper, we take advantage of a model-based method and extend it to three dimensional user-service-time tensor data. The proposed method is efficient in predicting the missing QoS values as

analyzed in Section III-D. Moreover, we constrain that the predicted value should not vary a lot from the average QoS value, which improves the prediction accuracy significantly.

VI. Conclusion and Future Work

Based on the intuition that a user's Web service QoS usage experience can be predicted by using the past usage experience from different users, we propose a novel model-based approach, called WSPred, for time-aware personalized QoS value prediction for Web services. By employing a collaborative framework, WSPred performs feature modeling on user, Web service and time based on the QoS usage experience collected from both local and global users. Requiring no additional invocation of Web services, WSPred makes the QoS prediction by evaluating how the user-specific, service-specific and time-specific latent features apply to each other. The extensive experimental results show that our proposed WSPred outperforms the state-of-the-art QoS prediction approaches for Web services.

For future work, we will investigate more techniques for improving the prediction accuracy (e.g., data smoothing, utilizing content information, etc.). Currently, we predict the values of different QoS properties independently. In the future, we will also conduct more investigations on the correlations and combinations on the different QoS properties. WSPred predicts missing QoS values based on the past QoS experience and the available QoS information in the current time interval. If no QoS information is available in the current time interval, WSPred purely depends on the past experience. In the future, we will explore an online prediction algorithm to perform time series analysis for prediction and extend WSPred to handle updated QoS information at run-time.

Acknowledgment

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 415311) and sponsored in part by the National Basic Research Program of China (973) under Grant No. 2011CB302600.

References

[1] http://en.wikipedia.org/wiki/Web_service.

[2] T. O reilly, "What is Web 2.0: Design patterns and business models for the next generation of software," *Communications and Strategies*, vol. 65, p. 17, 2007.

[3] J. El Haddad, M. Manouvrier, and M. Rukoz, "Tqos: Transactional and qos-aware selection algorithm for automatic web service composition," *IEEE Transactions on Services Computing*, pp. 73–85, 2010.

[4] T. Yu, Y. Zhang, and K. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, pp. 6–es, 2007.

[5] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *Proc. of WWW'10*, 2010, pp. 11–20.

[6] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proc. of WWW'09*, 2009, pp. 881–890.

[7] G. Canfora, M. Di Penta, R. Esposito, and M. Villani, "QoS-aware replanning of composite Web services," in *Proc. of ICWS'05*, 2005, pp. 121–129.

[8] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proc. of WSDM'10*, 2010, pp. 81–90.

[9] http://axis.apache.org/axis2/java/core.

[10] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[11] http://en.wikipedia.org/wiki/Cross validation.

[12] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together: Using socio-technical networks to predict failures," in *Proc. of ISSRE'09*, 2009, pp. 109–119.

[13] Z. Zheng and M. Lyu, "Collaborative reliability prediction of service-oriented systems," in *Proc. of ICSE'10*, 2010, pp. 35–44.

[14] M. Lyu *et al.*, "Handbook of software reliability engineering," 1996.

[15] M. Lyu, *Software fault tolerance*. John Wiley & Sons, 1995.

[16] Y. Zhang, Z. Zheng, and M. Lyu, "Wsexpress: A qos-aware search engine for web services," in *Proc. of ICWS'10*, 2010, pp. 91–98.

[17] Z. Zheng, Y. Zhang, and M. Lyu, "Distributed qos evaluation for real-world web services," in *Proc. of ICWS'10*, 2010, pp. 83–90.

[18] V. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Collaborative location and activity recommendations with gps history data," in *Proc. of WWW'10*, 2010, pp. 1029–1038.

[19] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Kim, P. Compton, and A. Mahidadia, "Learning collaborative filtering and its application to people to people recommendation in social networks," in *Proc. of ICDM'10*, 2010, pp. 743–748.

[20] Y. Shi, M. Larson, and A. Hanjalic, "Exploiting user similarity based on rated-item pools for improved user-based collaborative filtering," in *Proc. of Recsys'09*, 2009, pp. 125–132.

[21] W. Chen, J. Chu, J. Luan, H. Bai, Y. Wang, and E. Chang, "Collaborative filtering for orkut communities: discovery of user latent behavior," in *Proc. of WWW'09*, 2009, pp. 681–690.

[22] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[23] M. Jamali and M. Ester, "Trustwalker: a random walk model for combining trust-based and item-based recommendation," in *Proc. of KDD'09*, 2009, pp. 397–406.

[24] S. Gong, "A collaborative filtering recommendation algorithm based on user clustering and item clustering," *Journal of Software*, vol. 5, no. 7, pp. 745–752, 2010.

[25] Y. Zhang, Z. Zheng, and M. Lyu, "Exploring latent features for memory-based qos prediction in cloud computing," in *Proc. of SRDS'11*, 2011.

[26] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proc. of SIGIR'05*, 2005, pp. 114–121.

[27] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *Proc. of NIPS'08*, vol. 20, pp. 1257–1264, 2008.

[28] P. Singla and M. Richardson, "Yes, there is a correlation:-from social networks to personal behavior on the web," in *Proc. of WWW'08*, 2008, pp. 655–664.