PLMwsp: A Probabilistic Latent Model for Web Service QoS Prediction

Bobaker Mohamed A. Madi[†], Quan Z. Sheng[‡], Lina Yao[§], Yongrui Qin[*] and Xianzhi Wang[‡]
[†]Faculty of Information Technology, University of Benghazi, Libya; Email: Bobakermadi@yahoo.com
[‡]School of Computer Science, The University of Adelaide, Australia;
Email: {michael.sheng, xianzhi.wang}@adelaide.edu.au
[§]School of Computer Science and Engineering, UNSW, Australia; Email: lina.yao@unsw.edu.au
[*]School of Computing and Engineering, University of Huddersfield, United Kingdom; Email: yongrui.qin@hud.ac.uk

***Abstract*—With the unprecedented and dramatic development of Web services in recent years, designing novel approaches for efficient Web service prediction has become of paramount importance. Quality of Service (QoS) plays a critical role in Web service recommendation. However determining QoS values of Web services is still a challenging task. For example, some QoS properties (e.g., response time, throughput) may hold different values for different users. In this paper, we describe how to develop a novel approach, PLMwsp, based on a probabilistic latent model, to predict effectively the QoS values of Web services. A Web service prediction has been developed, and experiments have been conducted to show the efficacy of our approach.**

Keywords*— *Web Services, Quality of Service, Web Service Prediction, Probabilistic Latent Model.

I. Introduction

During the era of Web 2.0, Web service has been supported intensively by service providers due to the high demand from organizations and clients. Several providers have designed and produced a number of Web services for customers. Many customers used and liked these services, leading to dramatic increases in the number of individuals that use the Web. As a result, the Web has become more dynamic and attractive. Web service is software created to facilitate the interaction among devices and the Internet. It makes a considerable contribution for customers as well as providers. In addition, the unprecedented, dramatic development of Web services recently has called for effective approaches to Web service prediction and recommendation, which is an active issue in the area of service computing [1], [2] . The success of Web services has led to the concept of Quality of Service (QoS) [3]. There have been several approaches and frameworks that add QoS to Web services. Nowadays, providers are paying increasing attention to QoS. Recently, a number of publications on QoS have raised concerns regarding service-oriented computing and a number of QoS models have been implemented for improving Web service automatic composition and Web service selection [4], [5].

However, many service providers are likely to under-deliver QoS with respect to what is reported. In addition, some QoS values rely on terms like location and existence in the network. For the dataset we use, the distance between the user and the Web service is a factor that affects the value of QoS attributes, such as response time and throughput.

Evaluation of QoS of Web services can be conducted by the service user; however, such method has some disadvantages. First, it requires an invocation from the users, which means imposing costs on them. Second, in many situations, there are a considerable number of services to be evaluated, some of which are appropriate but undiscovered by the user. Furthermore, some service users lack the necessary experience to evaluate the services [6], [7].

Analyzing the data of each Web service would be tedious and consume much time. Thus, Web service QoS prediction techniques, which demand no additional invocations of the corresponding Web service, are becoming increasingly popular. Web service QoS prediction aims to make personalized QoS value predictions for service users by employing partially available information, such as characteristics of the current user, QoS information of the users, and historical QoS performance of Web services and users. To predict Web service QoS values with high accuracy, comprehensive investigations of the prediction approaches are needed.

There are three main methodologies to obtain Web service QoS values. First, QoS values can be obtained from the Web services providers, who could provide the QoS parameters. But this method is impractical, as the service providers may not deliver the QoS they declared, and some QoS properties are highly related to the locations and network conditions of the service users. Second, QoS values could be obtained from users' use experiences and evaluations of Web services, which may produce results that are more accurate. However, there is an obvious drawback in that it depends highly on users' situations. Different users at different locations or in different situations invoking the same services may experience quite different QoS results. Collaborative filtering-based (CF-based) techniques are widely used for uncovering user services dyadic interactions in QoS, such as using neighbourhood-based CF techniques to tackle problems that are likely to be faced by similar users or that originate from similarly delivered service items. Third, another method is to estimate QoS change by building up a model based on subjective factors (locations, network situation) affecting QoS performance. Subsequently, this model could be used to estimate the QoS values.

As such, to tackle the aforementioned downside, we propose a novel approach based on a probabilistic latent model. The basic idea of our method is to predict the values of Web services by using user observable variables and Web service observable variables, which will be modeled

978-1-5090-2675-3/16 $31.00 © 2016 IEEE
DOI 10.1109/ICWS.2016.86

in a matrix, called the user Web service matrix. In order to generate the final user Web service matrix, we will compute a latent model of users and services before combining them. In each step, we compute the noise variable by Gaussian distribution for both the observable model and the latent model. Then, we apply the stochastic gradient decent to find suitable theta values that can produce a predicted value with decreased error. Subsequently, we compare the prediction results with the real values in the experimental stage by applying mean absolute error (MAE) and root mean squared error (RMSE) to check the accuracy of our approach.

The rest of this paper is organized as follows: Section II includes a description of the related work and the approaches that have been proposed. Section III provides a presentation of our methodology for Web Service QoS prediction, followed by a data description and metrics that have used for assessing our methodology. Section IV is devoted on giving a brief description of the comparison approaches and an experimental evaluation on the performance of the proposed approach. Finally, Section V concludes this paper.

II. Related work

Researchers have recently begun attending to the importance of providing new techniques for predicting QoS attributes for Web services [8], [9], [6], [10], [11]. Many QoS attributes of Web service have been considered and they provide different feasibility and usability [12], [3], [13]. From these studies, response time, throughput, latency and availability are some commonly used attributes. Note that, in this paper, we concentrate on response time and throughput, which are the most import attributes in measuring QoS of Web services.

In the following, we review existing approaches for tackling the problem of predicting QoS attributes of Web services. In the literature, various techniques have been utilized for this purpose, including Multi Agents, collaborating filtering based approaches and Matrix Factorization Model [6], [14].

Malak *et al.* proposed an approach in 2009 based on Multi Agents techniques [14]. They designed a clear architecture of a prediction engine for a selection system. In terms of the agent system, each process can be included with a word agent. After a request from a consumer, they divide the process into stages. The stages are started by the consumer agent and then move to a selection agent where a prediction operation is performed.

In the area of selection and recommended systems, there are a number of systems using collaborative filtering for predicting unknown needed values [6], [9], [11], [15], [16], [17], [18], [19]. The collaborative filtering technique was coined by Goldberg et al. in 1992 [20]. They have proposed this technique for filtering information, namely for filtering email information. This concept is classified under the information filtering technique [21]. The most common technique of collaborative filtering is based on the neighbourhood approach. The first model of the neighbourhood approach is user-user based [22]. It is used for predicting the unknown values based on the recorded value of a similar user.

The idea of collaborative filtering has inspired many people who work on classification problems, particularly in areas of prediction, selection and recommender systems [23], [24]. For example, Zheng *et al.* have proposed an approach based on the collaborative filtering technique for a Web service recommender system [6]. They apply experiments on a dataset of QoS Web services by implementing a Pearson correlation coefficient (PCC) to conduct the comparison and determine the similarity between a service user and a Web service item.

Some related studies have also employed latent models in recommender systems [25] [10]. Zhang *et al.* proposed latent features for QoS prediction in cloud computing [10]. They design an algorithm called CloudPred to predict QoS values. The main algorithm depends on another algorithm called latent feature learning algorithm, which has been designed to correct the initial values of users and Web services. These values, which are randomly generated, are not highly accurate. Thus, to increase the accuracy, Zhang *et al.* generate the user-Web services matrix from the initial values of both users and Web service. Thereafter, they use the newly generated matrix with the user-Web services matrix of the real values (which comes from the dataset). The latent feature learning algorithm then uses the two matrices to correct the values of users and Web services. Once the values converge, the next step would be applying a similarity computation using the PCC. After applying the main algorithm, the missing QoS values can be predicted.

Salakhutdinov and Mnih proposed a probabilistic matrix factorization technique to perform on a large, sparse, and very imbalanced dataset [26]. However, their results were not very accurate, and they developed another idea by describing a fully Bayesian treatment of the probabilistic matrix factorization model. The distinguishing feature of their work is the use of Markov chain Monte Carlo methods for approximate inference in the new model.

Hybrid techniques have been proposed as well. The basic idea of hybrid techniques is to combine two or more techniques to benefit from the features in each technique [27]. Due to the disadvantages of some approaches, hybrid techniques have been used to provide a partial solution for addressing such drawback. For instance, it can be beneficial when we can combine an agent technique with a latent model or neighbourhood.

Tang *et al.* proposed a hybrid collaborative filtering technique in 2012 [28]. They concentrated on utilizing the distance between the locations of the user and Web service to predict the missing value. Tang *et al.* included two techniques, one for detecting similar users for an active user and another for finding similar services for a target service. They combine the two techniques into a hybrid collaborative filtering method.

Yao *et al.* proposed a novel approach that unifies collaborative filtering and content-based recommendations. In particular, their approach considers simultaneously both rating data (e.g., QoS) and semantic content data (e.g., functionalities) of Web services using a probabilistic generative model.

Yu *et al.* proposed in 2013 a combination approach for prediction of QoS Web services. They applied the idea of poor service (poor performance) for classifying services [7]. Their approach has three steps to predict QoS. The first step is finding the Top-K poor services using IMEAN. IMEAN is a metric that they designed for determining poor services and sorting services by order. The second step is if the predicted

service is one of the Top-K poor services, the metric is utilized for predicting its QoS performance. The third step is that they used an adjusted UPCC (a user-based prediction algorithm using PCC) for predicting the QoS performance when the predication service was not one of the Top-K poor services. For evaluating the results, they used the two metrics, mean absolute error (MAE), and its variation, normalized mean absolute error (NMAE).

III. Methodology

In this section, we describe our method in details.

A. Model Construction

Our task is predicting the QoS attributes of Web Services based on modeling the historical dyadic interactions between Users $\mathcal{I}$ and Services $\mathcal{J}$. Let the dyadic interactions from user $i \in \mathcal{I}$ and Web service $j \in \mathcal{J}$, interact with each other; their dyadic interactions can be denoted by$y_{ij} \in \mathcal{Y}$. Therefore, the mapping relationship can be denoted as follows:

$$\{(i,j) \rightarrow y_{ij}, i \in \mathcal{I}, j \in \mathcal{J}\} \tag{1}$$

This mapping creates a large matrix $Y \in \mathcal{Y}^{|\mathcal{I}|\times|\mathcal{J}|}$. However, this matrix is very sparse and noisy, and many entries are missing; therefore, our goal is to predict the value of any missing entry $\widehat{y_{ij}}$ given an incoming pair (i,j). The known interactions define a graph and the task amounts to propagating the sparse observations to the unobserved values of the matrix. For convenience, we will henceforth refer to i as the user and j as the Web Service item in this approach.

TABLE I. Important Notations

Symbol	Definition
$U = \{u_1, u_2, ..., u_i\}$	users' observable variables
$V = \{v_1, v_2, ..., v_j\}$	services items' observable variables
$\Phi = \{\phi_1, \phi_2, ..., \phi_i\}$	users' latent variables
$\Psi = \{\psi_1, \psi_2, ..., \psi_j\}$	services items' latent variables
$i \in \mathcal{I}$	users set
$j \in \mathcal{J}$	services items set
$\mathcal{Y} = \{y_{ij}\}$	historical QoS value when user i use services item j.
$\widehat{\mathcal{Y}} = \widehat{y_{ij}}$	predicted QoS value for the missing entry when user i use service item j
$\varepsilon = \{\varepsilon_y, \varepsilon_\phi, \varepsilon_\psi\}$	model parameters

To estimate y_{ij}, we can see two dependencies in our model: i) y_{ij} depends on both Users and Services items observable variables and latent variables $[y_{ij}|u_i, v_j, \phi_i, \psi_j, \varepsilon_y]$; ii) Users latent variables depend on their observable variables $[\phi_i|u_i, \varepsilon_\phi]$ and $[\psi_j|v_j, \varepsilon_\psi]$. Further clarification of the variables is shown in Table I.

To predict the dyadic interactions between Users and QoS Web Services, we need to construct a model representing the dependencies mentioned above in Figure 1. The users observable variables, U, can be generated by singular value decomposition (SVD) as shown in equation 4 and 5. The Users latent factors, ϕ_U, cannot be observed directly (e.g., the Users preference for QoS of a certain Web Service), but can be determined by the Users observable variables. In addition, the Web Service observable variable set, V, which denotes the service item textual information for its functional or non-functional description, together with the Users observable variables, can generate the value of Web Service observable variables from

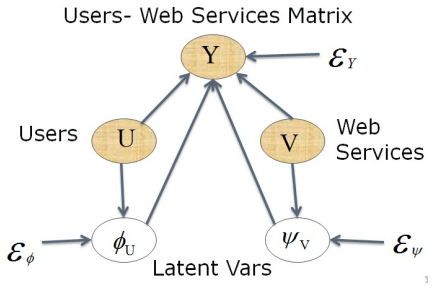

Fig. 1. Relationship between the variables

SVD. Thereby we can apply a learning algorithm to increase their accuracy. The latent factors of the service items follow the same discipline as those of the users, which cannot be observed directly, and ψ_V depends on the service items observable factors. Therefore, the model can be defined as follows:

$$\begin{aligned} Pr(\widehat{Y}) &\sim Pr(\widehat{Y}|U, V, \phi_U, \psi_V, \varepsilon) \\ where. Pr(\phi_U) &\sim Pr(\phi_U|U, A, \varepsilon_\phi) \\ Pr(\psi_V) &\sim Pr(\psi_V|V, B, \varepsilon_\psi) \end{aligned} \tag{2}$$

where $\varepsilon = \{\varepsilon_y, \varepsilon_\phi, \varepsilon_\psi\}$.

This approach is mainly based on a typical learning approach. From the known values of User u_i and Web Service v_j we can predict the missing values. The approach is simplified by a factorization technique so we can easily generate the values of *U* and *V*, and then implement the steps of our models. In this area of factorization, there are many approaches, including using a factorization-style collaborative filtering algorithm [29] [30].

The factorization for the User-Web Service Matrix (Y) it can be shown as the following:

$$SVD(Y) = S\Sigma D^*. \tag{3}$$

After factorization, the User-Web Service Matrix assumes the value of U is equal to $S\Sigma$ and that V is equal to D^*, as follows:

$$U = S\Sigma. \tag{4}$$

$$V = D^*. \tag{5}$$

To clarify the model shown above, we need to illustrate the relationship among the variables in three steps, which are as follows:

Step 1: Dependency between observable variables and latent variables

As described, we have a two-latent variable: one for Users and one for Web Services, as shown in the equations below. We assume that the latent features are the functions of observable variables in our model, and we assume this function is a linear function. The dependency can be defined as:

$$\phi_U = A \cdot U + \varepsilon_U \tag{6}$$

and

$$\psi_V = B \cdot V + \varepsilon_V \tag{7}$$

where ε_U and ε_V are the noise in this function, and we assume that both of them follow the Gaussian distribution, $\varepsilon_U \sim N(0,1)$ and $\varepsilon_V \sim N(0,1)$. In addition, A and B follow the Gaussian distribution $A \sim N(\mu_U, \sigma_U)$ and $B \sim N(\mu_V, \sigma_V)$.

To find the latent variables, we first find the values of User variable U and Web Services variable V. In our dataset, they are both vectors. This step is challenging, as other variables depend on the values of U and V and we should find appropriate values that converge quickly in our learning algorithm. In this case, we apply SVD on the matrix to get the observable variables. Thereafter we find variables A, B, ε_U and ε_V as shown in Equations 8, 9, 10 and 11.

As mentioned above, ε_U and ε_V are noise variables and follow Gaussian distribution with mean at zero (μ=0) and standard deviation value at one (σ =1).

$$\varepsilon_U = \mathcal{N}(U|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(U-\mu)^2/2\sigma^2} \tag{8}$$

and

$$\varepsilon_V = \mathcal{N}(V|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(V-\mu)^2/2\sigma^2} \tag{9}$$

Variable A follows Gaussian distribution, where the mean equals to the mean of U (μ_U), and the standard deviation equals to the standard deviation of U (σ_U).

$$A = \mathcal{N}(U|\mu_U, \sigma_U^2) = \frac{1}{\sqrt{2\pi\sigma_U^2}} e^{-(U-\mu_U)^2/2\sigma_U^2} \tag{10}$$

Variable B follows Gaussian distribution, where the mean equals to the mean of Web service V (μ_V), and the standard deviation equals to the standard deviation of Web service V (σ_V).

$$B = \mathcal{N}(V|\mu_V, \sigma_V^2) = \frac{1}{\sqrt{2\pi\sigma_V^2}} e^{-(V-\mu_V)^2/2\sigma_V^2} \tag{11}$$

After finding U and V and computing the other variables, A, B, ε_U and ε_V, from Gaussian distribution equations, we can compute the two latent variables (ϕ_U and ψ_V), as shown in Equations 6 and 7.

Step 2: Relevance for observable variables and latent variables

In our model, the observable variables and latent variables are not independent. For the relevance of observable variables, we assume observable features through a bilinear product, so that this dependency for the Users observable variables and Service items variables can be denoted as in Equations 8 and 9. This provides affinity between observations and corresponding dyadic interactions. For latent variable dependency, we inherit the latent factor model-based CF techniques, and the Users latent variables and Service items variables dependency can be assumed as a multiplicative model, which can be denoted as: $U^T V$ and $\phi_U^T \psi_V$.

Step 3: Specify the $\widehat{Y}$: After the first two steps, we can specify the $\widehat{Y}$; the results from Step 1 and Step 2 are a linear sum, which can be denoted as:

$$Pr(\widehat{Y}) \sim Pr(h_\theta(x_i)|u_i^T v_j + \phi_i^T \psi_j + \varepsilon_y), i = 1...n, and, j = 1...n. \tag{12}$$

where

$$h_\theta(x_i) = \theta_0 + \theta_1 e^{x_i} \tag{13}$$

The variable x denotes the index of Web Services. For instance, if a User has invoked 5825 Web Services, then any invoked Web service x_i would be any number between 1 and 5825.

B. Model Learning

This model is accomplished by computational framework. We fit it by using a traditional supervised learning approach, namely nonlinear regression and stochastic gradient descent in the optimization phase [31].

We use the computational model-based method, the optimization framework, to solve this model. It minimizes the $h_\theta(x_i) - y_{ij}$, where $h_\theta(x_i)$ is the predicted value between u_i and v_j. This can be rewritten as:

$$\min h_\theta(x_i) - U^T V + \phi_U^T \psi_V + \varepsilon_y \tag{14}$$

We adopt the L_2 loss function and L_2 regularization based on [31], so that we can define this equation:

$$\min \sum_{i \in \mathcal{I}, j \in \mathcal{J}} \ell(h_\theta(x_i), \widehat{y}_{ij}) + \sum_{i \in \mathcal{I}} \ell(U - \phi_U) + \sum_{j \in \mathcal{J}} \ell(V - \psi_V) + \frac{\lambda_a}{2}||A||^2 + \frac{\lambda_b}{2}||B||^2 \tag{15}$$

Where $\ell(\cdot)$ is the loss function, we adopt the least square function, which is: $\frac{1}{2}\sum_{i \in \mathcal{I}, j \in \mathcal{J}} \{h_\theta(x_i) - y_{ij}\}^2$. The $||\cdot||$ and coefficient λ. govern the relative importance of the regularization term compared with the sum-of-squares error term. The error functions can be minimized in a closed form. To solve the parameters, we adopt the stochastic gradient descent.

C. Non-Linear regression

Generally, the hypotheses function can be represented as a vector of θ's and a vector of features X. It shows as follows:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \qquad X = \begin{bmatrix} e^{x_0} \\ e^{x_1} \\ \vdots \\ e^{x_n} \end{bmatrix}$$

Because of the lack of features that should be included with the dataset, we use just one feature, x_1. As explained before, we make x_0 equal to 1 during the learning stage, and x_1 denotes the index of Web Service. As a result, the two vectors with two values can be written again as follows separately:

$$\theta = \begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix} \qquad X = \begin{bmatrix}e^{x_0} \\ e^{x_1}\end{bmatrix}$$

From the equation we need to transpose θ to be θ^T. The hypothesis can be rewritten again as shown below.

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i e^{x_i} = \theta^T e^x = [\theta_0\, \theta_1] \begin{bmatrix}e^{x_0} \\ e^{x_1}\end{bmatrix} \tag{16}$$

The equation of the hypotheses can be written as$h_\theta(x) = \theta_0 + \theta_1 e^x$. For simplicity, we need to represent the cost function by the symbol $J(\theta)$, as shown in Equation 17.

$$J(\theta) = \frac{1}{2}\sum_{i=0}^{n}(h_\theta(x_i) - y_i)^2 \tag{17}$$

D. Stochastic Gradient Descent

During model learning, we need to minimize Function 14, which is simplified in the cost function of non-linear regression in Equation 17. A suitable well-known technique is Stochastic Gradient Descent (SGD).

To minimize $J(\theta)$ we need to update θ in the main equation of gradient descent in each iteration which is:

$$\theta_j = \theta_j - \alpha \frac{d}{d\theta_j} J(\theta) \tag{18}$$

Where α is the step size. The term to the right of α is the derivative of the cost function $J(\theta)$. To simplify Equation 18 we need to apply derivative on it to obtain the final equation.

$$\frac{d}{d\theta_j} J(\theta) = \frac{d}{d\theta_j} \frac{1}{2}(h_\theta(x_i) - y_i)^2$$

$$\frac{d}{d\theta_j} J(\theta) = 2\, \frac{1}{2}(h_\theta(x_i) - y_i) \frac{d}{d\theta_j}(h_\theta(x_i) - y_i)$$

$$\frac{d}{d\theta_j} J(\theta) = (h_\theta(x_i) - y_i) \frac{d}{d\theta_j}(h_\theta(x_i) - y_i)$$

We need to substitute the hypotheses function $h_\theta(x_i)$ by its value $\sum_{i=0}^{n} \theta_i e^{x_i}$ to accomplish the derivative.

$$\frac{d}{d\theta_j} J(\theta) = (h_\theta(x_i) - y_i) \frac{d}{d\theta_j}(\sum_{i=0}^{n} \theta_i e^{x_i} - y_i)$$

$$\frac{d}{d\theta_j} J(\theta) = (h_\theta(x_i) - y_i)\, e^{x_i}$$

For a single training example, Formula 18 can be written again as in Equation 19. For clarity, we place the i between two brackets to show that i is not power.

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) e^{x_j^{(i)}} \tag{19}$$

IV. Experiments

In this section, we describe our experimental settings and present detailed performance reports of our approach.

A. Data description

We evaluated our model on a well-known dataset generated by Zheng[1][10]. This dataset was generated from 339 users and 5825 Web services and was saved in a massive matrix, namely, the user-item matrix, for response time and throughput. The rows in the matrix represent the users that who invoked Web services while the columns represent the Web service attributes. The size of the matrix is 339×5825, which means it has 1974675 invocations. In the response time matrix, there are a number of Web service invocation values assigned as -1. The value -1 indicates that the value could not be obtained for some reason, including connection refused or very long loading time [10]. Table II shows the statistics of the data.

TABLE II. Statistics of the Data

Statistics	Values
Number of Web Service	5825
Number of Service Users	339
Number of Web Service Invocation	1974675
Number of Web Service Countries	73
Number of User Countries	30
Mean of Throughput	102.86kbps
Mean of Response Time	1.43s
Standard Deviation of Throughput	531.85
Standard Deviation of Response Time	31.9s

B. Metrics

We evaluate PLMwsp by a metric widely used in collaborative filtering techniques, mean absolute error (MAE), and its variation, normalized mean absolute error (NMAE), to evaluate the performance of our algorithm. These two metrics are mainly for evaluating the predictive accuracy for the CF-based algorithms.

MAE calculates the average of the absolute difference between the predicted values and the real values:

$$MAE = \frac{\sum_{i,j} |\widehat{y_{ij}} - y_{ij}|}{n} \tag{20}$$

The other metric, *root mean squared error(RMSE)*, is defined as:

$$RMSE = \sqrt{\frac{\sum_{ij} (\widehat{y_{ij}} - y_{ij})^2}{n}} \tag{21}$$

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

where n is the total number of predicted QoS values, $\widehat{y_{ij}}$ is a predicted value, and y_{ij} is a real value. By applying 20 and 21, we examine our model and compare the results with other methods that utilized similar metrics. We use these metrics to assess PLMwsp by choosing many different matrix densities. For instance, if we assume that the density is 5%, the test results indicate the accuracy is very low.

C. Comparison approaches

1) CloudPred: The authors in CloudPred [10] generated the dataset that we used in our experiments. Basically, in their approach, they use a latent feature-learning algorithm for making the latent variables be learned until convergence is reached. After getting the values of the latent variables, they compute the similarity among all users by employing PCC. The PCC value falls into the interval [-1, 1]. If the PCC value is -1, there is no correlation between users; if the PCC value is +1, the two users have a strong correlation. In CloudPred, the authors prefer to exclude PCC with negative values due to their effect on prediction accuracy. For the prediction stage, they use the Top-K to predict the missing values for current users.

2) UPCC: The user-based collaborative filtering (UPCC) technique also uses the PCC. It employs the PCC to determine similarities between users and predicted QoS values based on identical users [23]. In the dataset that we work on, there is a massive matrix where the rows represent users. Applying this approach on that dataset would give rise to a comparison between each user (row) with all other users (all other rows). For instance, if we started from the first user (first row), then we would find all similar values with other users (other rows). In this case, the mechanism of choosing is clear. Thus, the user that has high similarity can be used to predict the missing QoS values of the active user.

3) IPCC: Item-based collaborative filtering (IPCC) is similar to UPCC. However, instead of focusing on users, it concentrates on items. IPCC also utilizes the PCC. The key point of using the PCC is to find similarity among items to predict QoS values according to these similar items [32], [33]. As mentioned above, the dataset that we have used in this project has a huge matrix. The columns of that matrix represent the items (Web service attributes). The same idea that is used in UPCC for comparison of the rows can be employed on the columns to find the similarity between the items (Web services attributes). As a result, it is easy to recognize the items with high similarity and to find the items that we need to predict the missing values.

4) UIPCC: UIPCC is a hybrid technique that takes advantages of UPCC and IPCC. UPCC focuses on behavior and characteristics of the user, while IPCC concentrates on the behavior and characteristics of the item itself. Ma *et al.* propose an approach that combines the two techniques to accomplish highly accurate missing value prediction [34]. The predicted values are based on a similar user and a similar item. When we apply this approach on the dataset in this research, the item will be the Web service attribute. UIPCC balances similar user values and similar Web service values to optimize the predicted values for the active user.

5) NMF: Non-negative matrix factorization (NMF) is proposed by Lee and Seung to handle user item datasets for predicting missing values [35]. It applies the non-negative matrix factorization technique for factorizing the user item matrix. In addition, they use principle component analysis and vector quantization in addition to NMF for holistic learning.

D. Experiments and Results

In this section, we report the prediction quality of the proposed PLMwsp technique with some state-of-the-art Web service recommendation techniques, including user-based CF, item-based CF, and latent factor model CF. We designed the experiments for evaluating the impact of different matrix densities on our algorithm's performance. The matrix density has affected the results in that as the density increases, we obtain better results. For instance, a density of 10% is much better than 5%, and a density of 20% is much better than 10%.

Our prediction algorithm aims at computing a ranking score for each candidate service item (which the user has not invoked) and returning the prediction value of this service to a targeted user. From the dataset, we apply the SVD to obtain the two observable variables, the user observable variable and the Web service observable variable. Thereafter, it is easy to generate the other two latent variables, the user latent variable and the Web service latent variable. As shown previously in the model, we can predict a missing value for an active user.

To evaluate prediction accuracy, we were interested in determining how many service items previously marked off in the pre-processing step were recovered in the returned services QoS prediction. By applying the learning technique, which is SGD, we can learn from non-active users who invoked Web services to predict the new values. We use different ratios of training data to evaluate all the methods, e.g., we randomly select 20% of QoS values as the training dataset to predict the rest of QoS values (e.g., 80%).

Table III presents the comparison between the all approaches, including PLMwsp. We employ the comparison just for 4 cases of matrix density due to the availability of data in [10]: 10%,20%,80% and 90%. To assess the result, we compare them with the results of the other approaches. If the value of MAE and RMSE in each specific density for any approach has the highest value, then that approach has the worst value. As a result, the prediction value will have low accuracy. For instance, from table III, IPCC has the highest value, which means that IPCC is the worst approach. In contrast, an approach with lower values of MAE and RMSE is a better approach. For example, PLMwsp has the lowest value of MAE and RMSE, which means that our approach has highest accuracy for predicted values. Figures 2 visualizes the numerical results of the comparison. The red line represents the results of our model (PLMwsp).

In Figure 2, we compare the prediction accuracy of all approaches under different matrix densities. We start from 10% and move to 90%, with a step value of 10%. Figures 2(a) and (b) show the values of all methods that applied on the response time matrix, while figures 2(c) and (d) present the values of all approaches that implemented throughput. The X-axis represents the density values and the Y-axis represents the metric type. As the density of the matrix increases, the

TABLE III. PERFORMANCE COMPARISON.

Matrix Density	Metrics	Response Time (Seconds)						Throughput (kbps)					
		IPCC	UPCC	UIPCC	NMF	CloudPred	**PLMwsp**	IPCC	UPCC	UIPCC	NMF	CloudPred	**PLMwsp**
10%	MAE	0.7596	0.5655	0.5654	0.6754	0.5306	**0.0905**	31.6722	26.2015	22.6567	19.7700	19.0009	**17.2954**
	RMSE	1.6133	1.3326	1.3309	1.5354	1.2904	**1.0172**	65.5220	61.9658	57.4653	57.3767	51.8236	**27.2155**
20%	MAE	0.7624	0.5516	0.5053	0.6771	0.4745	**0.0881**	35.1780	21.9313	18.1230	15.7794	15.4203	**15.3723**
	RMSE	1.6257	1.3114	1.2486	1.5241	1.1973	**1.0168**	66.6028	56.5441	50.0435	50.1402	44.8975	**26.4091**
80%	MAE	0.6703	0.4442	0.3873	0.3740	0.3704	**0.0658**	29.9146	14.5497	12.4880	12.5107	10.7881	**3.8257**
	RMSE	1.4102	1.1514	1.0785	1.1242	1.0597	**1.0130**	64.3079	44.3738	39.6017	39.2029	36.8506	**18.3626**
90%	MAE	0.6687	0.4331	0.3793	0.3649	0.3638	**0.0455**	29.9404	13.8761	12.0662	11.6960	10.4722	**1.9091**
	RMSE	1.4173	1.1264	1.0592	1.1121	1.0359	**1.0093**	63.7149	42.5534	38.0763	36.7555	35.9225	**15.3438**

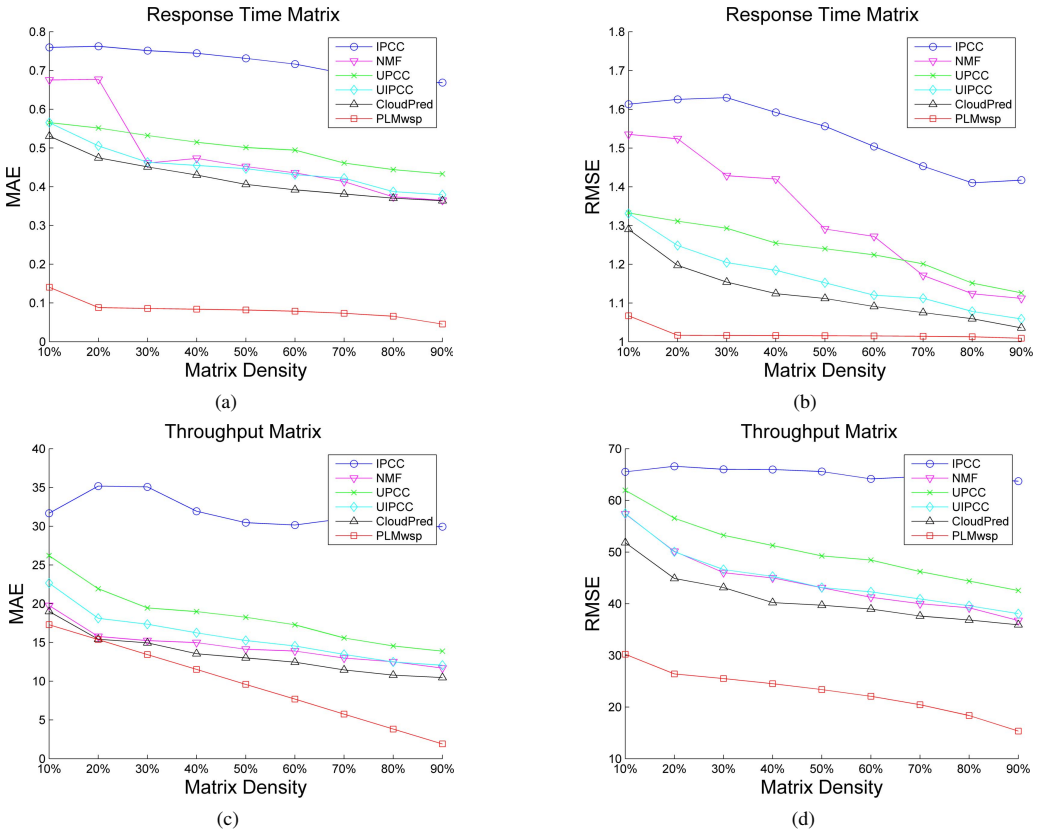

Fig. 2. (a) Impact of matrix density by using MAE on response time (b) Impact of matrix density by using RMSE on response time (c) Impact of matrix density by using MAE on throughput, (d) Impact of matrix density by using RMSE on throughput

line in the figures decreases. The reason behind this is that when the density matrix increases, it means more informative QoS values are invloved as training data for prediction, the error decreases accordingly.

Actually, the NMF performs better than IPCC and UIPCC in most cases (see Table III and Fig.2) except in a single case shown in Fig. 2(b). It should be understood that a better algorithm may occasionally perform worse than an ordinary one. Apparently, Both NMF and UIPCC utilize more heuristics than IPCC. Therefore, their overall performance is theoretically better.

From the previous results in the tables and figures that we obtained for PLMwsp, we can see that our model achieves the best efficacy when we compare the results with other approaches. Non-linear regression plays an essential role for improving the results of PLMwsp. We can apply PLMwsp to other Web service attributes apart from response time and throughput in a similar way.

V. CONCLUSION

In this paper, we have proposed a probabilistic latent model (PLMwsp) for predicting QoS values of Web services. The objective of PLMwsp is to predict the values of Web service QoS by using a user observable variable and a web service observable variable. We used SVD to generate the two variables from the matrix-user web service matrix. Then, we computed the latent variables of users and services by generating them from the observable variables. We obtained on the needed matrix by applying three computation steps. We evaluate PLMwsp through a set of experiments and comparisons with existing approaches. The proposed PLMwsp has been demonstrated that it can provide higher efficacy and quality of prediction than other approaches.

REFERENCES

[1] L.-J. Zhang, J. Zhang, and H. Cai, *Services computing*. DE: Springer Verlag, 2008.

[2] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web Services Composition: A Decade's Overview," *Information Sciences*, vol. 280, pp. 218–238, 2015.

[3] A. Al-Moayed and B. Hollunder, "Quality of service attributes in web services," in *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, aug. 2010, pp. 367 –372.

[4] J. El Hadad, M. Manouvrier, and M. Rukoz, "Tqos: Transactional and qos-aware selection algorithm for automatic web service composition," *Services Computing, IEEE Transactions on*, vol. 3, no. 1, pp. 73–85, 2010.

[5] 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*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 881–890. [Online]. Available: http://doi.acm.org/10.1145/1526709.1526828

[6] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *Proceedings of the 2009 IEEE International Conference on Web Services*, ser. ICWS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 437–444. [Online]. Available: http://dx.doi.org/10.1109/ICWS.2009.30

[7] D. Yu, M. Wu, and Y. Yin, "A combination approach to qos prediction of web services." Springer Berlin Heidelberg, 2013, pp. 99–106. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37804-111

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

[9] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "Collaborative web service qos prediction with location-based regularization," in *Proceedings of the 2012 IEEE 19th International Conference on Web Services*, ser. ICWS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 464–471.

[10] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based qos prediction in cloud computing," in *Proceedings of the 2011 IEEE 30th International Symposium on Reliable Distributed Systems*, ser. SRDS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–10.

[11] Z. Zheng, Y. Zhang, and M. Lyu, "Distributed qos evaluation for real-world web services," in *Web Services (ICWS), 2010 IEEE International Conference on*, 2010, pp. 83–90.

[12] Y. Zhang, Z. Zheng, and M. Lyu, "Wsexpress: A qos-aware search engine for web services," in *Web Services (ICWS), 2010 IEEE International Conference on*, 2010, pp. 91–98.

[13] L. Yao, Q. Z. Sheng, A. H. Ngu, J. Yu, and A. Segev, "Unified collaborative and content-based web service recommendation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 453–466, 2015.

[14] J. S. Malak, M. Mohsenzadeh, and M. A. Seyyedi, "Web service qos prediction based on multi agents," in *Proceedings of the 2009 International Conference on Computer Technology and Development - Volume 01*, ser. ICCTD '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 265–269.

[15] K. Karta, "An investigation on personalized collaborative filtering for web service selection, technical report, available online at http://citeseerx.ist.psu.edu/ viewdoc/summary?doi=10.1.1.90.6961 (last accessed on april 2013)," 2005.

[16] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Major components of the gravity recommendation system," *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 80–83, Dec. 2007. [Online]. Available: http://doi.acm.org/10.1145/1345448.1345466

[17] J. Ge, Z. Chen, J. Peng, T. Li, and L. Zhang, "Web service recommendation based on qos prediction method," in *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, 2010, pp. 109–112.

[18] H. Sun, Z. Zheng, J. Chen, and M. Lyu, "Nrcf: A novel collaborative filtering method for service recommendation," in *Web Services (ICWS), 2011 IEEE International Conference on*, 2011, pp. 702–703.

[19] H. Liu, F. Zhong, and B. Ouyang, "A web services selection approach based on personalized qos prediction," in *Parallel and Distributed Computing (ISPDC), 2011 10th International Symposium on*, 2011, pp. 199–206.

[20] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992. [Online]. Available: http://doi.acm.org/10.1145/138859.138867

[21] J. Cohen, "Special issue on information filtering," *Communciations of the ACM*, vol. 35, no. 12, 1992.

[22] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 1, pp. 1:1–1:24, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1644873.1644874

[23] 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*, 2007, pp. 439–446.

[24] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 447–456. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557072

[25] D. M. Titterington, "Some aspects of latent structure analysis," in *Proceedings of the 2005 international conference on Subspace, Latent Structure and Feature Selection*, ser. SLSFS'05. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 69–83. [Online]. Available: http://dx.doi.org/10.1007/117527904

[26] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 20, 2008.

[27] J. Cao, Z. Wu, Y. Wang, and Y. Zhuang, "Hybrid collaborative filtering algorithm for bidirectional web service recommendation," *Knowledge and Information Systems*, pp. 1–21, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10115-012-0562-1

[28] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*, 2012, pp. 202–209.

[29] Y. Fang and L. Si, "Matrix co-factorization for recommendation with rich side information and implicit feedback," in *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, ser. HetRec '11. New York, NY, USA: ACM, 2011, pp. 65–69. [Online]. Available: http://doi.acm.org/10.1145/2039320.2039330

[30] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.263

[31] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[32] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, ser. CSCW '94. New York, NY, USA: ACM, 1994, pp. 175–186. [Online]. Available: http://doi.acm.org/10.1145/192844.192905

[33] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, ser. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 43–52. [Online]. Available: http://dl.acm.org/citation.cfm?id=2074094.2074100

[34] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 39–46. [Online]. Available: http://doi.acm.org/10.1145/1277741.1277751

[35] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization." Macmillan Magazines Ltd., 1999, pp. 788–791.