PERSONALIZED QOS PREDICTION USING TIME SERIES FORECASTING AND COLLABORATIVE FILTERING

Yan Hu\textsuperscript{1,2}, Qimin Peng\textsuperscript{2}, Xiaohui Hu\textsuperscript{2}, and Rong Yang\textsuperscript{3}
\textsuperscript{1}University of Chinese Academy of Sciences, Beijing, China
\textsuperscript{2}Science and Technology on Integrated Information System Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
\textsuperscript{3}College of Computer Science and Technology, Hubei University of Science and Technology, Xianning, China
\{huyan, qimin, hxxh\}@iscas.ac.cn, harry804@163.com

Abstract
Quality of Service (QoS) has been widely used for web service recommendation and selection. Since QoS information usually cannot be predetermined, how to make personalized QoS prediction precisely becomes a big challenge. Time series forecasting and Collaborative Filtering (CF) are two mainstream technologies for QoS prediction. However, existing time series forecasting approaches based on AutoRegressive Integrated Moving Average (ARIMA) models do not take new observations as a feedback to revise QoS forecasts. Moreover, they only focus on forecasting QoS values for each individual web service. Service users’ personalized factors are not considered. CF facilitates personalized QoS evaluation, but rarely models the temporal dynamics of QoS values. To address the limitations of existing approaches, this paper proposes a novel personalized QoS prediction approach considering both the temporal dynamics of QoS attributes and the personalized factors of service users. Our approach seamlessly combines CF with an improved time series forecasting method, which uses Kalman filtering to compensate for defects of ARIMA models. Additionally, we design a prototype system for QoS dissemination over the Internet, thus providing a necessary infrastructure for the implementation of personalized QoS prediction. Finally, we conduct experiments to study the effectiveness of our approach. Experimental results show our approach can improve QoS prediction accuracy significantly.

Keywords: Personalized QoS prediction, web service, ARIMA, Kalman filtering, collaborative filtering.

1. INTRODUCTION
Web services provide a standardized solution for Service-Oriented Architecture (SOA)\textsuperscript{[1]}, which satisfies the requirements of standards-based, loosely-coupled, and protocol-independent distributed computing. They encapsulate application functionalities and are identified by URIs which can be advertised, located and accessed through messages encoded according to XML-based standards like SOAP and WSDL\textsuperscript{[2]}, thus allowing interaction and collaboration between different machines over a network. The exponential growth of web services on the Internet calls for effective web service selection\textsuperscript{[3]–[5]} and recommendation\textsuperscript{[6]–[8]} technologies. And with the proliferation of functionally equivalent web services, we need additional information to distinguish them.

Quality of Service (QoS)\textsuperscript{[9]} describes the non-functionalities of web services, including response time, throughput, time between failures, etc. It plays an important role in differentiating web services which provide similar functionalities\textsuperscript{[10]}. However, QoS performance of web services usually cannot be predetermined and service providers rarely deliver the QoS as declared due to:

- Local QoS performance of web services has a high volatility over time in such a dynamic Internet environment. Varying work load and varying number of concurrent accesses inevitably affect local QoS performance of web services.
- User-side QoS observation is not only based on the local performance of a web service, but also influenced by some personalized factors, like network connections, link lengths and users’ geographical positions. Different users may observe completely different QoS values even if they invoke a same web service.

Therefore, how to effectively integrate both the time-varying and personalized features of QoS values for QoS prediction is of great importance for high-quality web service recommendation and selection.

To capture the temporal dynamics of QoS, time series forecasting based on ARIMA models\textsuperscript{[11]–[13]} is regarded as a useful technique for QoS prediction. It mainly utilizes AutoRegressive Integrated Moving Average (ARIMA) models to fit the historical QoS data and then to forecast future QoS values. This technology is popular due to its simplicity and usability. However, its limitations should not be ignored. ARIMA models cannot correct forecasts timely due to its inability to take the new observation as a feedback. Moreover, to the best of our knowledge, none of the existing QoS series forecasting approaches gives attention to users’ personalized requirements. They only focus on forecasting future QoS values for each individual web service, which is not enough. QoS performance of web services observed by users is highly influenced by users’ personalized factors. Different users may observe totally different QoS performance even if
they invoke a same web service. As for personalized user-side QoS evaluation, neighborhood-based Collaborative Filtering (CF) [14], [15], [16], [17], [18], [19] is a very popular prediction technique. Since personalized QoS observation data is usually sparse, it is difficult to construct a time series model for each pair of user and service to make personalized QoS series forecasting. Fortunately, CF approaches are capable of taking full advantage of historical QoS information from similar neighbors (users [14], [15], services [16], [17], or both [18], [19]) to predict missing QoS values for a target user on a group of candidate web services which meet the functional requirements. However, very few CF-based approaches have the capacity to precisely characterize the temporal dynamics of QoS. Accordingly, designing effective mechanisms to integrate both the two important facets into user-side QoS prediction is meaningful to personalized web service recommendation and selection.

In this paper, we propose a novel personalized QoS prediction approach, which seamlessly integrates two phases: local QoS forecasting for each individual web service based on time series analysis and CF-based user-side personalized QoS prediction. The first phase utilizes Kalman filtering to compensate for the limitations of ARIMA models. The second phase employs a modified neighborhood-based CF algorithm to make user-side QoS prediction.

The key contributions of our work include:

- An improved time series forecasting approach integrating ARIMA models with Kalman filtering is proposed to capture the temporal dynamics of QoS and forecast future QoS values for each individual web service. Kalman filtering can effectively make up for the deficiency of ARIMA models by utilizing its feedback mechanism to correct forecasts timely.

- A modified neighborhood-based CF algorithm takes both the QoS series forecasting results and the historical QoS information from similar neighbors as inputs to make user-side personalized QoS prediction, thus to achieve high-quality web service recommendation and selection.

- A prototype system, which is based on a popular message middleware ActiveMQ, is designed for QoS dissemination between different clients over a network, thus to build a necessary infrastructure for the implementation of personalized QoS prediction. The system is implemented by Java language and deployed on the Internet.

- Several experiments are conducted to study the effectiveness of our proposed personalized QoS prediction approach.

The rest of this paper is organized as follows. Section 3 states the problem and presents an overview of the proposed approach. Section 4 elaborates on how to utilize ARIMA model combined with Kalman filtering to forecast local QoS performance, and then make personalized QoS prediction based on QoS series forecasting results and a modified neighborhood-based CF algorithm. In Section 5 we design and implement a prototype system for QoS dissemination over Internet. Afterwards, several experiments are conducted to evaluate the proposed approach in Section 6. Finally, conclusions are drawn in Section 7.

2. RELATED WORK

QoS prediction has been regarded as an important technique to support QoS management and QoS-based web service recommendation and selection in dynamic environments. In research literature, there are two mainstream technologies for QoS prediction: time series forecasting and CF-based approaches.

Time series forecasting approaches have been successfully applied to modeling and forecasting QoS data. They use different models to fit the past QoS values and then forecast their future changes. Zhu et al. [20] employ a QoS series model which involves three parts, a trend item, a periodical item and a random item, to characterize the linearity, periodic and uncertain changes of QoS values and use this model to extrapolate the series into the future. Li et al. [21] propose a time series model based on structural equations to fit QoS measures of web services, and predict their future values dynamically. Godse et al. [12] propose a method that combines monitoring technologies with extrapolation methods based on ARIMA models to predict future service performance. Amin et al. [22] present an improved QoS forecasting approach which integrates ARIMA with GARCH models to address the constant variation assumption limitation of ARIMA models. Cavallaro et al. [11] conduct an empirical study aimed at comparing different QoS prediction approaches based on time series forecasting, namely the adoption of average and current values, linear models, and ARIMA models. They conclude that ARIMA forecasting has the best compromise in being sensible to outliers, being able to predict possible violations of QoS constraints, and ensuring a low forecast error.

Although ARIMA models have prominent advantages, they still have some defects which should not be ignored. First, they cannot correct forecasts timely by taking account of the latest QoS observations. Second, ARIMA forecasting only focuses on predicting future QoS values for each individual web service. Users’ personalized factors are not taken into account. The actual QoS observations by different users are not only affected by the local QoS performance of web services but also related to users’ personalized factors (e.g., network conditions, link lengths and physical locations). Therefore, we should also consider users’ personalized factors while making user-side QoS prediction.
CF approaches are popular techniques for personalized QoS prediction, which are mainly divided into two categories, neighborhood-based [14], [15], [16], [17], [18], [19] and model-based [23], [24], [25]. Neighborhood-based CF utilizes historical QoS information from different users or services to measure their similarities on personalized QoS experiences. Then, historical QoS information from similar neighbors is collected to make personalized QoS prediction for a target user on a set of candidate web services. The most analyzed neighborhood-based CF approaches include user-based [14], [15], service-based [16], [17] and hybrid [18], [19] methods. The user-based method utilizes the historical QoS experiences from similar users for personalized QoS prediction, while the service-based method uses information from similar services for prediction. The hybrid method is the combination of the previous two methods, so it can achieve higher prediction accuracy. Model-based CF mainly utilizes training data to train a predefined model (e.g., clustering models [25], aspect models [24] and latent factor models [23]) and then employs the trained model for QoS prediction. Neighborhood-based CF is preferred than model-based CF since it is more intuitive to interpret QoS prediction results.

Some CF algorithms try to deal with the temporal dynamics of QoS values. Works [26], [27] and [28], which belong to the neighborhood-based CF category, employ empirical weights, e.g., a simple average or an exponentially weighted average, to evaluate the joint impacts of historical QoS information at various time points on QoS prediction. However, it is still hard to capture the temporal dynamics of QoS values precisely only with those empirical techniques. Another two works [29] and [30], belonging to the model-based CF category, accomplish time-aware QoS prediction by formalizing the problem as user-service-time tensor factorization models. However, tensor factorization will become intractable when the tensor size becomes large. Moreover, it is a common defect of model-based CF approaches that they cannot interpret QoS prediction results intuitively.

From these existing works, we can conclude that the temporal dynamics and users’ personalized factors are two key facets of QoS prediction. However, whether time series forecasting or collaborative filtering is only good at solving one aspect. Therefore, it is necessary to design effective mechanisms to integrate the two kind of methods, utilizing their respective advantages to compensate for their limitations, thus to achieve more accurate personalized QoS prediction.

3. PROBLEM DEFINITION AND APPROACH OVERVIEW

This section presents the problem definition and a concise introduction to our innovative personalized QoS prediction approach.

3.1 Problem Definition

As stated above, QoS prediction has been widely used for web service recommendation and selection. Here, for simplicity, we describe the QoS prediction problem in the context of web service recommendation, and it can be easily applied to other contexts with small changes. Suppose that a web service recommender system contains total M users \( U = \{ u_1, u_2, \ldots, u_M \} \) and N web services \( S = \{ s_1, s_2, \ldots, s_N \} \). Each service has its local QoS performance changing over time, as shown in Figure 1. When a user invokes a web service, he observes the QoS information of this service from his own perspective, namely the user-side QoS observation. We employ a three-dimensional user-service-time matrix \( Q \) to denote the relationships between users and services, as shown in Figure 2. If user \( u_i \) (\( u_i \in U \)) invoked service \( s_j \) (\( s_j \in S \)) during the time interval \( t_k \) (\( k = 1, 2, \ldots \)), the observed QoS value is recorded in the entry \( q_{i_k,j} \) of matrix \( Q \), otherwise the corresponding entry is called a missing QoS value.

After receiving the specified functional requirements from a target user \( u_i \), who requests web service recommendation at time \( t_{treq} \), the system should identify all candidate web services which satisfy the required functionalities and recommend one with optimal QoS performance to the target user. Accordingly, the missing QoS value should be predicted for each pair of \(( u_i, s_j )\)
(s_j is one of the candidate web services) at time t_{current}, taking into account the local performance of s_j and the historical user-side QoS information contained in Q.

3.2 Approach Overview
As shown in Figure 3, our approach mainly includes three procedures: 1) Build an ARIMA model for each individual web service to fit its historical local QoS performance, mainly including determining the model orders and estimating the model parameters by means of a set of well-defined calibration methods; 2) Seamlessly convert the constructed ARIMA model to a state space representation as required by Kalman filtering, and use this state space model to correct QoS forecasts timely by incorporating new QoS observations into the forecasting process; 3) Employ a modified neighborhood-based CF algorithm for user-side personalized QoS prediction, based on the time series forecasting results and the historical QoS observations provided by neighbors who have similar personalized QoS experiences. In the following sections, each procedure of our approach will be introduced in detail.

4. PERSONALIZED QoS PREDICTION APPROACH

4.1 ARIMA Models
ARIMA is originally proposed by Box and Jenkins [31], which is a very popular model for time series analysis and forecasting [32], [33]. It collects past values of the same variable and develops a model to describe their underlying relationships. Then the model is used to extrapolate the time series into the future. A time series \{x_t\} can be modeled by an ARMA model of orders p (for AR) and q (for MA), denoted by ARMA(p, q), if it satisfies:

\[ x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t, \]

where \( x_t \) and \( x_{t-i} \) (i = 1, 2, ..., p) are the current and past stationary observations, respectively, \( \varepsilon_t \) and \( \varepsilon_{t-j} \) (j = 1, 2, ..., q) are the current and past random errors which are assumed to be serially independent and normally distributed with a mean of zero and a variance of \( \sigma^2 \). \( \phi_i \) (i = 1, 2, ..., p) and \( \theta_j \) (j = 1, 2, ..., q) are model parameters.

If the original time series \( x_t \) is non-stationary, \( d \) differences should be done to transform it into a stationary one \( \hat{x}_t \). Consequently, \( x_t \) is said to be generated by an ARIMA model of orders \( p, d, q \), denoted by ARIMA(p, d, q). Box and Jenkins stated that a time series should satisfy serial dependency and stationarity and ARIMA models should satisfy stationarity and invertibility [31]. Examining whether these assumptions are satisfied is a critical task, as unsatisfied assumptions generate improper ARIMA models which in turn lead to biased forecasts.

4.2 ARIMA Model Construction for QoS Series
An ARIMA model is constructed based on the collected QoS observations through the following phases:

P1) Data Preparation: Before constructing an ARIMA model, we should check whether the original time series data satisfy the underlying assumptions described above. If they are not satisfied, we should find proper transformations to make the time series fulfill these assumptions.
We can approximately determine the orders of the MA model and the AR model, respectively. It is worth noting that the approach does not identify only one adequate model, but identifies a combination of adequate models based on the dependency structure of the given QoS series data.

Example ▶ We use a benchmark QoS data set [11] of nine real-world web services, which were invoked for about four months. Information about these web services is shown in Table 1. Response time (RT) of each service is computed, by measuring the time taken between sending a request to a service and receiving a response. For the data set WS1(RT), the time series assumptions are tested as follows: 1) Serial dependency: a correlation diagram and Ljung-Box Q-statistics shown in Figure 4a are utilized to check whether the QoS series WS1(RT) is mutually independent.

We can see that WS1(RT) is serially dependent due to the fact that the 1st, 2nd, 3rd-order autocorrelations and the 1st-order partial autocorrelation are significantly different from zero, i.e., overstepping the bounds of 2 standard deviation (two dashed lines), and the p-values of all Q-statistics are smaller than 0.05, which means the null hypothesis that the QoS series is independent should be rejected; 2) Stationarity: the approach uses ADF test to check the stationarity of the QoS series. The stationarity is satisfied because the p-value equals to 0.000 (< 0.05), which indicates that we ought to reject the null hypothesis the QoS series is non-stationary.

\[ P2) \text{ Model Identification:} \] After data preparation, this phase selects the most suitable orders \(p, d, q\) for the ARIMA\((p, d, q)\) model. The observation of autocorrelation function (ACF) and partial autocorrelation function (PACF) of the QoS series can help to make this selection. If the ACF curve decays and the PACF curve cuts off, AR models are adequate to characterize the prepared data. If the ACF curve cuts off and the PACF curve decays, MA models are adequate. Otherwise, if both ACF and PACF decay or cut off, ARMA models are adequate. The significantly non-zero orders of ACF and PACF approximately determine the orders of the MA model and the AR model, respectively.

![Figure 4. ACF and PACF of WS1(RT) and residuals](image-url)

<table>
<thead>
<tr>
<th>WS Id</th>
<th>WS Name</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS1</td>
<td>XML Daily Fact</td>
<td>Returns a daily fact with an emphasis on XML Web Services and the use of XML</td>
<td><a href="http://www.xmlme.com/WSDailyXml.asmx">http://www.xmlme.com/WSDailyXml.asmx</a></td>
</tr>
<tr>
<td>WS2</td>
<td>Amazon</td>
<td>Searches for books in the online book store</td>
<td><a href="http://www.xmlme.com/WSAmazonBox.asmx">http://www.xmlme.com/WSAmazonBox.asmx</a></td>
</tr>
<tr>
<td>WS3</td>
<td>BLiquidity</td>
<td>Provides information on liquidity in a banking system</td>
<td><a href="http://webservicex.com/BLiquidity/BLiquidity.asmx">http://webservicex.com/BLiquidity/BLiquidity.asmx</a></td>
</tr>
<tr>
<td>WS4</td>
<td>Currency Converter</td>
<td>Performs a currency conversion using the current quotation</td>
<td><a href="http://www.webservicex.com/CurrencyConverter.asmx">http://www.webservicex.com/CurrencyConverter.asmx</a></td>
</tr>
<tr>
<td>WS5</td>
<td>Fast Weather</td>
<td>Reports weather info for a given city</td>
<td><a href="http://ws2.serviceobjects.net/fw/FastWeather.asmx">http://ws2.serviceobjects.net/fw/FastWeather.asmx</a></td>
</tr>
<tr>
<td>WS6</td>
<td>GetJoke</td>
<td>Outputs a random joke</td>
<td><a href="http://www.interpressfast.net/webservicex/getJoke.asmx">http://www.interpressfast.net/webservicex/getJoke.asmx</a></td>
</tr>
<tr>
<td>WS7</td>
<td>Hyperlink Extractor</td>
<td>Extracts hyperlinks from web pages</td>
<td><a href="http://www.atomicx.com/xmlservices/HyperlinkExtractor.asmx">http://www.atomicx.com/xmlservices/HyperlinkExtractor.asmx</a></td>
</tr>
<tr>
<td>WS8</td>
<td>Quote of the Day</td>
<td>Reports a random quote every day</td>
<td><a href="http://www.swanandmokashi.com/HomePage/WebServices/QuoteOfTheDay.asmx">http://www.swanandmokashi.com/HomePage/WebServices/QuoteOfTheDay.asmx</a></td>
</tr>
<tr>
<td>WS9</td>
<td>Stock Quote</td>
<td>Reports quotations of stocks</td>
<td><a href="http://www.xmlme.com/WSDailyXml.asmx">http://www.xmlme.com/WSDailyXml.asmx</a></td>
</tr>
</tbody>
</table>

Table I. Characteristics of the Monitored Real Services.


$ d = 0 $ is already determined due to the stationarity of $ WS(RT) $.

**P3) Model Estimation:** In this phase, parameters of identified models should be estimated to provide the best fit to the prepared QoS series data. Maximum likelihood estimation (MLE) is a popular method for parameter estimation [31]. Model parameters are estimated by maximizing a likelihood function for the available QoS series data. For explanation, we assume that $ \{x_1, x_2, \ldots, x_n\} $ is a QoS series of the ARIMA model (1) with $ \{\varepsilon_t\} \sim \text{Normal}(0, \sigma^2) $. The likelihood function $ l $, i.e., the joint probability of total $ n $ observations, is denoted by:

$$ l \propto \left(\sigma^2\right)^{-n/2} \exp \left\{-\frac{1}{2\sigma^2} \sum_{t=1}^{n} (\varepsilon_t)^2 \right\}, $$

where $ \varepsilon_t = x_t - \sum_{i=0}^p \phi_i x_{t-i} - \sum_{j=1}^q \theta_j \varepsilon_{t-j} $. Values of the parameters $ \phi_i $ and $ \theta_j $ maximizing the likelihood function $ l $ are referred to as the maximum likelihood estimates.

Example $ \triangleright $ After identifying a group of tentative ARIMA models in $ P2 $, the approach uses MLE method for parameter estimation for each model. The estimates are listed in Table 2. $ \triangleright $

**P4) Model Checking:** Model checking involves the diagnostic checking for model adequacy. If at least one diagnostic is not satisfied, the current model is inadequate and should be removed from the set of tentative models identified in $ P2 $. The primary diagnostics are estimates significance, stationarity, invertibility and residuals randomness.

Example $ \triangleright $ The approach computes the $ t $-statistics for all the estimates to investigate their significance, as shown in Table 2. We can conclude that only the estimates of the first, the second and the last models are significant, because the $ p $-values of the three models are smaller than 0.05, which means the null hypothesis that estimates are nonsignificant should be rejected. Additionally, all the models with significant parameters satisfy stationarity and invertibility, as the roots of $ \Phi(z) = 1 - \sum_{i=0}^p \phi_i z^{-i} = 0 $ and $ \Psi(z) = 1 + \sum_{j=1}^q \theta_j z^{-j} = 0 $ all lie outside the unit circle.

To analyze the randomness of residuals, we compute their autocorrelations, partial autocorrelations and Q-statistics. As an example, the residuals of the last model is shown in Figure 4b. It is evident that the residuals are a random series, since all the correlation and partial correlation coefficients are close to 0, i.e., within the range of $ \pm 2 $ standard deviation (two dashed lines), and the $ p $-values of all Q-statistics are greater than 0.05 remarkably, which indicates the null hypothesis that the residual series is independent should be accepted. Similarly, the residuals of the first and second models also meet the randomness requirement. $ \triangleright $

**P5) Model Selection:** Once all candidate models are estimated and checked, the best model is selected based on Akaike’s Information Criterion (AIC):

$$ AIC = 2k - 2\ln(l) $$

where $ k = p + q $ is the number of parameters and $ l $ is the maximized value of the likelihood function of the estimated ARIMA model. It has been proved that

$$ \ln(l) \approx -\frac{n}{2} \ln(\sigma^2) + K, $$

where $ n $ is the length of the sample series, $ \sigma^2 $ is variance of $ \{\varepsilon_t\} $ and $ K $ is an inessential value which can be ignored. At the beginning, with the increase of $ p $ and $ q, AIC $ decreases. However, due to the limited length of the sample series, too large $ p $ and $ q $ lead to imprecise estimation, so $ \sigma^2 $ increases and accordingly $ AIC $ increases. Therefore, the model with the minimum $ AIC $ value is the best model.
Moreover, both \( \mathbf{w}_t \) and \( \mathbf{v}_t \) are independent, zero-mean, Gaussian noise processes with normal probability distributions: \( p(\mathbf{w}) = N(0, \mathbf{Q}) \) and \( p(\mathbf{v}) = N(0, \mathbf{R}) \), where \( \mathbf{Q} \) and \( \mathbf{R} \) are the covariance matrices of \( \mathbf{w}_t \) and \( \mathbf{v}_t \), respectively.

\[ P_{10} = A_{10} x_0 + B_{10} w_1, \]

where \( A_{10} \) is the state transition matrix, which relates the state \( x_t \) at step \( t \) to the state \( x_{t+1} \) at step \( t+1 \), \( B_{10} \) is the excitation transfer matrix relating the process noise \( w_1 \) to the state \( x_{t+1} \), \( H_{1} \) relates the state \( x_{t+1} \) to the measurement \( z_{t+1} \), and \( v_t \) is the measurement noise.

\[ z_{t+1} = H_{t+1} x_{t+1} + v_{t+1}, \]

where \( A_{t+1} \) is the state transition matrix, which relates the state \( x_t \) at step \( t \) to the state \( x_{t+1} \) at step \( t+1 \), \( B_{t+1} \) is the excitation transfer matrix relating the process noise \( w_{t+1} \) to the state \( x_{t+1} \), \( H_{t+1} \) relates the state \( x_{t+1} \) to the measurement \( z_{t+1} \), and \( v_t \) is the measurement noise.

The measurement update equations can be thought of as corrector equations. Once the actual QoS
measurement is observed, the forecasting process should be corrected as follows:

\[
K_{r+1} = P_{r+1}H_{r+1}^T \left[ H_{r+1}P_{r+1}H_{r+1}^T + R_{r+1} \right]^{-1},
\]

(13)

\[
x_{r+1} = x_{r+1} + K_{r+1} \left[ z_{r+1} - H_{r+1}x_{r+1} \right],
\]

(14)

\[
P_{r+1} = \left[ I - K_{r+1}H_{r+1} \right]P_{r+1},
\]

(15)

where \( R_{r+1} \) is the covariance matrix of process noise \( w_{r+1} \), \( I \) is the unit matrix, and \( K_{r+1} \) is the Kalman gain matrix, which is obtained by minimizing the posteriori covariance matrix \( P_{r+1} \) of prediction error, and \( P_{r+1} \) in Equation (14) is the corresponding minimized posteriori covariance matrix at step \( t+1 \). As shown in Equation (14), the posteriori state estimate \( x_{r+1} \) at step \( t+1 \) is obtained by linearly combining the priori state estimate \( x_{r+1} \) and a weighted difference between the actual measurement \( z_{r+1} \) and its prediction \( H_{r+1}x_{r+1} \). \( K_{r+1} \) determines how much \( x_{r+1} \) relies on the new measurement \( z_{r+1} \). If the past estimation is better (\( P_{r+1} \) smaller, then \( K_{r+1} \) smaller), the new measurement will be considered less (\( K_{r+1} \) smaller). During the implementation of Kalman filtering, the initial conditions can take empirical values: \( x_0 = [0] \), \( P_0 = 10I \), \( Q = I \) \( (t = 1,2,3,\ldots) \) and \( R = I \) \( (t = 1,2,3,\ldots) \). Kalman filtering utilizes the new measurement to correct the state estimate at each step. When the forecasting time is long enough, the impacts of initial \( x_0 \) and \( P_0 \) on forecasting results decay to 0.

4.4 Personalized QoS Prediction

The above time series forecasting method just forecasts local QoS performance for each individual web service. However, it is inadequate for user-side personalized QoS prediction. Personalized QoS observation is composed of two parts. One is the local QoS performance of each web service presented on the machine where the service is deployed. The other is the personalized part of each user. The first part is related to some local factors of a web service (e.g., the number of concurrent accesses, load and device conditions), while the second part is affected by users’ personalized factors, like network conditions, link lengths and users’ geographical locations. The personalized QoS observation can be seen as the additive effect of the two parts. For the first part, we employ the proposed time series forecasting method. Next, we expatiate on how to predict the personalized part.

**P9) User Similarity Measurement:** Some users may share similar personalized QoS experiences, as they may be in a same network or have the same links lengths to one or more web services. We use user similarity to measure the proximity between the personalized QoS experiences of any two service users. Supposing that \( u_1 \) and \( u_2 \) are two users, their similarity can be calculated by:

\[
sim(u_1, u_2) = \frac{\sum_{a \in S(u)} \left( q_{a, u_1} - q_{a, u_2} \right)}{\sum_{a \in S(u)} \left( q_{a, u} - q_{a, u} \right)},
\]

(16)

where \( S = S_{u_1} \cap S_{u_2} \) is the set of web services invoked by both \( u_1 \) and \( u_2 \), \( q_{a, u_1} \) is personalized QoS value observed by user \( u_1 \) on service \( s \) at time \( t \), \( q_{a, u_2} \) is the personalized QoS observation by \( u_2 \) on service \( s \) at time \( t' \), and \( q_{a, u_1} \) and \( q_{a, u_2} \) are the monitored local QoS values of service \( s \) at time \( t \) and \( t' \), respectively. It is worth noting that \( \left( q_{a, u_1} - q_{a, u_2} \right) \) and \( \left( q_{a, u_1} - q_{a, u_2} \right) \) are the respective personalized parts of QoS observations by \( u_1 \) and \( u_2 \) on \( s \). The above equation often overestimates user similarities, since two users may not be similar actually but happen to have similar personalized QoS experiences on a small number of co-invoked web services, so we use a similarity weight proposed in [18], [19] to reduce the impact of a small number of co-invoked web services:

\[
sim'(u_1, u_2) = \frac{2|S_{u_1} \cap S_{u_2}|}{|S_{u_1}| + |S_{u_2}|} \cdot \sim(u_1, u_2),
\]

(17)

where \( |S_{u_1}| \) and \( |S_{u_2}| \) are the numbers of web services invoked by \( u_1 \) and \( u_2 \), respectively, and \( |S_{u_1} \cap S_{u_2}| \) is the number of their co-invoked web services.

**P10, P11) Personalized QoS Prediction Based on Collaborative Filtering:** After computing user similarities, we utilize historical QoS information from similar users to make personalized QoS prediction. Suppose that \( u \) is a target user, who requests web service recommendation at time \( t_{current} \) and \( s \) is one of the candidate web services. All similar users of \( u \) form such a user set:

\( S(u) = \{ a \in U \mid \sim'(a, u) > 0 \} \). Here, only users whose similarities with \( u \) are greater than 0 can be regarded as his similar neighbors. Thus, the personalized part \( \hat{q}_p(\hat{u}, s) \) observed by user \( u \) on service \( s \) can be predicted by the following equation:

\[
\hat{q}_p(\hat{u}, s) = \frac{\sum_{a \in S(u)} \sim'(a, u) \left( q_{a, s,t} - q_{a,t} \right)}{\sum_{a \in S(u)} \sim'(a, u)},
\]

(18)

where \( q_{a,s,t} \) is the personalized QoS value observed by user \( a \) on service \( s \) at time \( t \), and \( q_{a,t} \) is the monitored local QoS performance of service \( s \) at time \( t \). As mentioned previously, the user-side QoS observation is the
additive effect of the local performance of a web service and the personalized part of a user. Therefore, the comprehensive personalized QoS prediction result \( \hat{q}(u,s,t_{current}) \) for user \( u \) on service \( s \) can be obtained by calculating the sum of the predicted personalized part \( \hat{q}_p(u,s) \) and the local QoS forecasting result \( \hat{q}_{current} \) of service \( s \):

\[
\hat{q}(u,s,t_{current}) = \hat{q}_p(u,s) + \hat{q}_{current}.
\]

(19)

5. A Prototype System for QoS Dissemination

The implementation of the personalized QoS prediction approach calls for effective and efficient QoS dissemination technology in an Internet environment. It should enable QoS information sharing between different clients, including users and web services distributed on the Internet. In this section, a prototype system for QoS dissemination over a network is designed. It is implemented with JDK, MyEclipse, Apache Tomcat and ActiveMQ. ActiveMQ\(^1\) is a lightweight open source message broker, which is designed for real-time communication between more than one client or server. The information transmission mechanism of ActiveMQ is illustrated in Figure 6. Real-time messages can be published on different topics, and information published on a topic will be pushed to all the subscribers of this topic with no delay. Each topic can publish messages from one or more publishers and can also be subscribed by one or more subscribers. Undoubtedly, ActiveMQ is an excellent message-oriented middleware supporting QoS dissemination between distributed clients on the Internet. Figure 7 shows the architecture of our prototype system, in which three kinds of components are involved, publishers, subscribers and a central server (or servers).

Publishers are responsible for collecting local QoS data and publishing them to the central server. Each user client or web service can act as a QoS publisher. In user side, a publisher should record QoS data observed by this user. In web service side, the publisher should keep track of its local QoS performance and forecast its future QoS values, i.e., local QoS forecasting for each web service is performed locally. If a client wants to be a publisher, it ought to firstly send a connection request to the central server. Then the server creates and allocates a topic name for this publisher. After successful connection, the publisher can release its QoS information to the corresponding topic. Thus, all subscribers who have subscribed to this topic can receive real-time QoS information published on this topic. If an online publisher wants to disconnect from the central server, it should send a disconnection request to the server. Then the server reclaims the topic name and dismisses the subscriber queue of this topic.

Subscribers are QoS information receivers. Every user client can become a QoS subscriber. Since web services don’t need to know QoS information from other clients, they just act as QoS publishers. User clients can choose to subscribe QoS data they are interested in, i.e., QoS information from candidate web services and similar neighbors. If a user client wants to become a subscriber, it should firstly send a connection request to the central server and tell the server what publishers it wants to subscribe to. Then the subscriber will receive a response of connection success and a series of requested topic names (one name related to one publisher) from the central server. In the meanwhile, the central server adds it to the subscriber queues of the corresponding topics. After that, the subscriber can successfully receive real-time QoS information published on those subscribed topics. If a subscriber wants to stop receiving information from

\(^{1}\)http://activemq.apache.org/
one topic, it should notify the server and will be removed from the subscriber queue of this topic.

Central Server is a controller for QoS dissemination, which has 3 main duties. The first is to manage the publish/subscribe relationships as described above. In more detail, it allows multiple QoS publishers and subscribers to connect to itself, and creates and allocates topics for QoS publishers, and manages the subscriber queue for each topic. The second is utilizing user-side QoS observations to perform user similarity computation (i.e., one part of the CF algorithm) and informing each user who are similar to him. Then, each user can subscribe to the QoS data published by his similar neighbors. Based on the subscribed information from similar neighbors and candidate web services, personalized QoS prediction can be implemented locally in user side. The third duty of the central server is to store historical QoS information released by different publishers. The stored QoS data can be used for future research and data analysis.

### 6 EXPERIMENTS

In this section, we conduct several experiments to study the effectiveness of our personalized QoS prediction approach.

#### 6.1 Study Setup

First, we identify the main research questions of our study as follows:

**Q1**: Does the proposed time series forecasting method which seamlessly combines ARIMA models with Kalman filtering improve the QoS forecasting accuracy comparing to ARIMA model or the ARIMA-GARCH model proposed in Work [22]?

**Q2**: Does the proposed personalized QoS prediction approach, which integrates time series forecasting with CF, outperform the traditional neighborhood-based CF algorithm [18], [19] and our previous Time Aware and Data Sparsity Tolerant CF approach (TADST) [27], [28]?

To address Q1, we apply our improved time series forecasting method, ARIMA model and ARIMA-GARCH model to the response time (RT) and throughput (TH) data sets of nine real-world web services provided in [11]. And we use Mean Absolute Percentage Error (MAPE) to measure the forecasting accuracy, which is defined by:

$$MAPE = \frac{1}{K} \sum_{k=1}^{K} \left| \frac{q_k - \hat{q}_k}{q_k} \right| \times 100\%,$$

where $q_k$ and $\hat{q}_k$ are the actual and predicted QoS values, respectively, and $K$ is the total number of predicted values. A smaller MAPE value indicates a higher forecasting accuracy. It is worth noting that the ARIMA-GARCH model is not always applicable to time series forecasting, only when dependency exists between the conditional variance of prediction error and past errors or the lags of the conditional variance [37]. Moreover, we measure the Relative forecasting Accuracy Improvement using a metric $RAI_{MAPE}$ defined by:

$$RAI_{MAPE} = \frac{MAPE_{ARIMA(GARCH)} - MAPE_{Ours}}{MAPE_{ARIMA(GARCH)}} \times 100\%, \quad (21)$$

where $MAPE_{Ours}$ is the MAPE value of the improved time series forecasting approach proposed in this paper, and $MAPE_{ARIMA(GARCH)}$ is the MAPE value of the ARIMA model or the ARIMA-GARCH model when necessary. A larger $RAI_{MAPE}$ means a higher accuracy improvement.

To address Q2, we still utilize MAPE to measure prediction accuracy of different approaches. Unfortunately, existing publicly available data sets, e.g., WS-DREAM and PW [38], only involve user-side QoS evaluations but lack local QoS time series data of each web service, so we have to build a synthetic data set to evaluate our approach. It includes 500 services and 500 users. For each service, we randomly generate a 300-point time series as its local RT series. For each pair of $(u, s_j)$ $(1 \leq i, j \leq 500)$, we generate a personalized RT series by performing the following two steps. First, we design a specific normal distribution function for user $u_i$ and use this function to generate the personalized part for this user on service $s_j$. Here, we assume the personalized parts of a user on different services conform to a specific normal distribution. Then, the personalized RT series for $(u_i, s_j)$ is generated by adding the personalized part to the local RT series of service $s_j$.

Based on the synthetic personalized RT data set, we build a $500 \times 500 \times 299$ user-service-time train matrix (excluding the last time interval) and a $500 \times 500$ (the last time interval) user-service test matrix. For each pair of $(u_i, s_j)$, we select a value during a random time interval $t_k \ (1 \leq k \leq 299)$ from its corresponding personalized RT series as the value of the $(i, j, k)$th entry of the train matrix, and the other entries, i.e., $(i, j, l)$ th $(l \neq k)$ are set to NULL. Moreover, the last time interval is thought of as the recommendation time $t_{\text{recent}}$, so the value during the last time interval of each personalized RT series is regarded as a test value and inserted into the corresponding entry of the test matrix.

Afterwards, we randomly remove some entries of the training matrix to make the density of training data vary from 10 percent to 100 percent, with a step value of 10 percent, to simulate different densities of training data. Additionally, the $TopK$ most similar neighbors are chosen for each target user. A neighbor should be removed from the set of $TopK$ similar neighbors if his similarity with the

---

2http://www.wsdream.net/dataset.html
6.2 Results

Q1. Improving forecasting accuracy comparing to ARIMA or ARIMA-GARCH models: In this experiment, we discover that only the QoS series of WS$_2$, WS$_3$ and WS$_4$ need the ARIMA-GARCH model to describe the dynamic changes of their time-varying conditional variance. For the other web services (i.e., WS$_1$, WS$_5$ ~ WS$_7$ and WS$_8$), ARIMA model is adequate. We present a graphical view of prediction accuracy comparison of ARIMA-Kalman (i.e., the improved time series forecasting approach in this paper), ARIMA and ARIMA-GARCH on WS$_2$(RT). The first 200 observations of WS$_2$(RT) are used for model estimation, and the one-step ahead forecasts of the total 400 observations by ARIMA, ARIMA-GARCH and ARIMA-Kalman are depicted in Figures 8a to 8c, respectively. Obviously, our approach produces a prediction curve matching the actual observation curve better than both ARIMA and ARIMA-GARCH models.

Boxplots of absolute percentage errors of different approaches are shown in Figure 9, while MAPE values of ARIMA-Kalman and ARIMA (or ARIMA-GARCH) are reported separately for each service in Table 3. The boxplots highlight that the proposed ARIMA-Kalman approach can provide more accurate forecasts than ARIMA (or ARIMAGARCH) model, since the prediction errors produced by ARIMA-Kalman are evidently lower than those of ARIMA (or ARIMA-GARCH) model. 50 percent of the prediction errors are within (6.15 percent, 30.00 percent) for ARIMA-Kalman, and within (9.03 percent, 31.95 percent) for ARIMA (or ARIMA-GARCH). And the median values of absolute percentage errors produced by ARIMA-Kalman and ARIMA (or ARIMA-GARCH) are 15.53 percent and 19.00 percent, respectively. More precisely, from Table 3, we can see that ARIMA-Kalman produces MAPE $= 17.87$ percent comparing to MAPE $= 22.24$ percent produced by ARIMA (or ARIMA-GARCH) with $R_{MAPE} = 19.65$ percent. And for each web service, it is evident that ARIMA-Kalman outperforms the ARIMA (or ARIMA-GARCH). The lowest and highest accuracy improvements of ARIMA-Kalman comparing to ARIMA (or ARIMA-GARCH) are in the cases of WS$_2$(TH) with $R_{MAPE} = 0.59$ percent and WS$_4$(TH) with $R_{MAPE} = 50.92$ percent.

Q2. Improving personalized QoS prediction accuracy comparing to traditional CF (WSRec) and our previous time aware and data sparsity tolerant CF approach (TADST): Prediction accuracy comparison of the proposed personalized QoS prediction approach, WSRec and TADST under different training data densities and different $TopK$ values are illustrated in Figures 10 and 11, respectively.
We clearly see that the proposed personalized QoS prediction approach significantly outperforms WSRec and TADST. We ascribe the higher-accuracy of our approach to the ability of time series forecasting methods to precisely characterize the temporal dynamics of QoS values, and our approach successfully integrates it with a modified CF algorithm. Although TADST also deals with time factors in QoS prediction, it is still hard to precisely capture the temporal dynamics of QoS values just with empirical techniques. This result shows another observation that when the values of training data density and TopK are smaller, TADST yields higher prediction accuracy than WSRec, but when the two values become larger, the advantage of TADST degrades. This is because TADST is more suitable to deal with data sparsity problem than WSRec. However, our proposed approach takes full advantage of local QoS information of web services, which can provide another excellent solution for traditional data sparsity problem. When there is no user-service-time historical QoS data available (the most extreme case of data sparsity), our approach can omit the personalized part and use local QoS forecasting results as personalized QoS prediction results, while TADST and WSRec cannot yield any prediction result. Therefore, with the help of local QoS information, no matter whether the user-service-time training matrix is sparse or not, our proposed approach can achieve much higher QoS prediction accuracy compared with existing CF approaches which only rely on the user observed QoS data. Consequently, our approach is more robust to handle data sparsity problem.

7 CONCLUSIONS

In this paper, we propose a novel personalized QoS prediction approach to support high-quality web service recommendation and selection. This approach utilizes an improved time series forecasting method, which takes advantage of the feedback mechanism of Kalman filtering to compensate for traditional ARIMA model, to model the temporal dynamics of local QoS performance of web services. Then, our approach seamlessly combines a modified neighborhood-based CF algorithm with the local QoS series forecasting method, to capture personalized features of user-side QoS observations. Moreover, in order

![Figure 10. Performance Comparison Under Different Training Data Densities](image1)

![Figure 11. Performance Comparison Under Different TopK values](image2)

| TABLE III: MAPE and RAI\_MAPE Values for the Predictions |
|-----------------|-----------------|-----------------|
| **WS\_id** | **QoS** | **MAPE Values(%)** | **RAI\_MAPE Values(%)** |
| | | **ARIMA-Kalman** | **ARIMA-GARCH** | **ARIMA** | **ARIMA** |
| WS\_1 | RT | 14.72 | 21.05 | 30.07 |
| | TH | 10.08 | 14.32 | 29.61 |
| WS\_2 | RT | 9.96 | 16.23 | 16.70 | 38.63 |
| | TH | 17.09 | 17.18 | 17.67 | 0.52 |
| WS\_3 | RT | 32.46 | 38.67 | 42.00 | 16.06 |
| | TH | 20.08 | 22.14 | 23.07 | 9.30 |
| WS\_4 | RT | 5.47 | 10.62 | 48.49 |
| | TH | 3.74 | 7.62 | 50.92 |
| WS\_5 | RT | 29.05 | 37.37 | 22.26 |
| | TH | 20.21 | 20.33 | 0.59 |
| WS\_6 | RT | 14.90 | 17.58 | 15.24 |
| | TH | 18.32 | 20.46 | 10.46 |
| WS\_7 | RT | 24.53 | 32.31 | 24.08 |
| | TH | 15.73 | 16.66 | 5.58 |
| WS\_8 | RT | 20.27 | 33.95 | 36.05 | 40.29 |
| | TH | 25.81 | 27.86 | 28.43 | 7.36 |
| WS\_9 | RT | 25.82 | 31.78 | 18.75 |
| | TH | 13.38 | 14.10 | 5.11 |
| **Average** | | 17.87 | 22.24 | 19.65 |
to provide a necessary infrastructure for the implementation of the personalized QoS prediction approach, we design and implement a prototype system for effective and efficient QoS dissemination over the network with Java language and deploy it to Internet. However, the work presented in this paper still has some limitations. The relatively small real-world data set and the synthesized data set used for experiments are vulnerable to biases, which may lead to unfair evaluation. Additionally, due to our hypothesis that users process services individually, the important impacts from service dependencies and usage patterns on personalized QoS prediction are omitted. These remaining issues will be taken into consideration in our future work.

8. ACKNOWLEDGEMENT

The authors appreciate the reviewers for their extensive and informative comments and suggestions for the improvement of this paper. The work described in this paper is supported by the Chinese 863 project (Project No. 2012AA011206) and the Chinese Scholarship Council (CSC).

9. REFERENCES


**Authors**

**Yan Hu** received the BEng degree in automation from Xi’an Jiaotong University, Xi’an, China, in 2011. She is currently working toward the PhD degree in the Science and Technology on Integrated Information System Laboratory of Institute of Software, Chinese Academy of Sciences and University of Chinese Academy of Sciences. Her main research interests include web service computing and information system integration.

**Qimin Peng** received the PhD degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2005. He is currently an Associate Professor and Master supervisor in the Science and Technology on Integrated Information System Laboratory, Institute of Software, Chinese Academy of Sciences. His main research interests are in web service computing, information system integration, image processing and pattern recognition.

**Xiaohui Hu** received the BEng degree from the Dalian University of Technology, Dalian, China, in 1982; the MEng degree from National University of Defense Technology, Changsha, China, in 1985; and the PhD degree from the Beihang University, Beijing, China, in 2003. He is currently a Professor, PhD supervisor and Assistant Chief Engineer in the Institute of Software, Chinese Academy of Sciences. He is also a director of the Science and Technology on Integrated Information System Laboratory. He was with Beijing Institute of Systems Engineering as an Associate Professor and deputy director from 1985 to 1997. From 1997 to 2005, he was a Professor of Beijing Institute of Systems Engineering and the director of its simulation laboratory. From 2005 to 2007, he was with the Institute of Software, Chinese Academy of Sciences. His research interests include web service computing, system simulation and information system integration.

**Rong Yang** received the BEng degree from the College of Computer Science at South-Central University for Nationalities in 2002; the MEng degree from the College of Mathematics and Computer Science at Wuhan Textile University in 2009; and the PhD degree from the Wuhan University in 2015. His main research interests include web service computing and software engineering.