Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 2153027, 14 pages
https://doi.org/10.1155/2019/2153027

Research Article

Runtime Quality Prediction for Web Services via Multivariate Long Short-Term Memory

Ling Guo,[1] Ping Wan,[1] Rui Li,[1] Gang Liu,[2] and Pan He[2]

[1]*Army Logistics University of PLA, Chongqing 401331, China*
[2]*Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China*

Correspondence should be addressed to Pan He; hepan@cigit.ac.cn

Received 27 March 2019; Revised 12 July 2019; Accepted 15 July 2019; Published 21 August 2019

Academic Editor: Anna Vila

Online quality prediction helps to identify the web service quality degradation in the near future. While historical web service usage data are used for online prediction in preventive maintenance, the similarities in the usage data from multiple users invoking the same web service are ignored. To improve the service quality prediction accuracy, a multivariate time series model is built considering multiple user invocation processes. After analysing the cross-correlation and similarity of the historical web service quality data from different users, the time series model is estimated using the multivariate LSTM network and used to predict the quality data for the next few time series points. Experiments were conducted to compare the multivariate methods with the univariate methods. The results showed that the multivariate LSTM model outperformed the univariate models in both MAE and RMSE and achieved the best performance in most test cases, which proved the efficiency of our method.

1. Introduction

As a major concern in service-oriented computing, the runtime quality or reliability assurance has aroused attentions from many researchers [1, 2]. Since the service-oriented systems usually operate in highly complicated and dynamic environments, predictive maintenance is employed as a main approach for reliability assurance [3]. This approach focuses on predicting service performance degradation and preventing the occurrence of failure with the help of service replacement or recomposition [4]. As to reduce the unnecessary cost related to each recomposition, maintenance should be performed when the failure is very likely to occur. So, it is useless to conduct the predictive analysis on service quality for a very long time. Instead, the quality prediction should be conducted in a runtime manner. Thus, a key requirement in the whole process is the accurate, real-time prediction of the performance of each invoked service in the near future [5]. While it is essential to conduct runtime service quality prediction, there are several kinds of unsolved challenges.

First of all, the quality of a web service often changes irregularly over time, which makes it difficult to identify the time-dependent regularities in it. To conduct runtime prediction, time-dependent models need to be built on top of the observation data. A lot of methods, such as ARIMA and semi-Markov model, are used for time series data prediction. However, they generally rely on statistical analysis of historical records and could not handle the nonstationary data as well as methods such as recurrent neural networks.

Secondly, existing methods on time series models leverage time-dependent historical web service usage data from a single user and employ univariate time series models. Measured from the user side, the service quality may be affected by the user's network or machine delay, which introduces uncertainty to the dataset. The efficiency of the prediction method may be affected by the random uncertainty. Since a web service is open to the public, multiple users may pay visit to the service in the same period of time. To reduce the random error from the service quality data and to improve the prediction accuracy, the similarities in the usage data of a single web service from multiple users during the same period could be taken into consideration.

Thirdly, while the performance of a component system is usually measured by resource usage, environment interactions,

possible faults, seasonal activities related to time, and etc., it is difficult to collect these measures solely from the client side. Although many specific issues related to system behaviour are comprised in event and performance logs, the historical execution and log information are also difficult to obtain from the service invokers. For black-box service environment, the quality prediction mainly relies on the easy-collected quality information from the invokers themselves.

So, most existing prediction methods are not suitable for the runtime service quality prediction. This paper aims to build a time series-based web service quality prediction model. Considering the above issues, this model intends to improve the short-time quality prediction accuracy in the open and dynamic service-oriented computing environment, only employing the quality data collected from client side. The improved model takes advantage of the historical quality data from the invocation process of multiple users. While the historical invocation data from one user is represented as a univariate time series model, the data of multiple invocation processes to the same service can be characterized by a multivariate time series model. On top of this assumption, a multivariate web service runtime quality prediction method is proposed accordingly. To improve the model accuracy, the historical quality data from multiple users are chosen through the correlation and similarity analysis. Since the frequent changes exist in the quality data, the parameters in multivariate time series model are solved using a LSTM network instead of statistical methods. An online algorithm is proposed employing this model for runtime service quality prediction in the near future. Experimental evaluations are conducted to compare the prediction accuracy between the univariate time series prediction methods and the multivariate method and to prove the efficiency of our method.

The rest of this paper is organized as follows. Section 2 lists the related work and the shortcomings. Section 3 provides the formulation of the runtime quality prediction problem and the multivariate time series model. Section 4 illustrates the solution to the model and the corresponding prediction algorithm. The experimental evaluations and results are shown in Section 5, followed by threats to validity in Section 6. Finally, conclusions and future work are given in Section 7.

2. Related Work

In order to enhance service-oriented system dependability, predictive maintenance is often used, including failure prediction and early malware detection [3]. While the whole predictive analytics process consists of several steps, the main aim of this paper only focuses on the prediction model. The studies of traditional software system reliability or performance analysis are mainly based on two approaches: the modelling-based approach and the measurement-based one. The former one models the system architecture or the system's running state together with failure occurrence so as to derive the system reliability or cost in the transient and steady state [6]. The latter one focuses on the detection and validation of symptoms (i.e., side effects of errors) that point to an upcoming failure using monitored data. It also focuses on the estimation of failures or the effects on system resources according to the current system state [7].

Since the service quality is often measured by the historical feature data, the measurement-based method is used for the quality prediction. Silic et al. considered specific parameters for users, services, and the environment and built cluster-based reliability prediction methods [8]. Considering the invocation context, e.g., service workloads and network conditions, Zhu et al. built a context-aware reliability prediction approach, taking advantages of the past web service usage experiences from users [9]. While the service quality often changes irregularly over time, these researches ignored the effect of time change.

In order to support timely service recomposition, the quality prediction should be conducted in a nearly real-time manner. The constructed stationary models, such as semi-Markov model, Bayesian networks support vector machines, and component interaction graph [10, 11], do not fully describe the frequent changes in the observation quality data over time. To identify time-dependent regularities in system reliability, some state-of-the-art approaches rely on statistical analysis of the time distribution of historical records [12]. The adoption of one single probabilistic distribution with certain inconsistency might cause errors in the future prediction, while the hybrid distribution introduces extra evaluation overhead [13, 14].

Time series analysis also makes predictions by analysing the characteristics of the system's historical parameters [15, 16]. Considering frequent quality changes in service-oriented systems over time, statistical time series models such as ARIMA are built to predict future quality of web services at the next time series point [17]. Using statistical time series models, each subsequent prediction depends on the previous predicted result and it is difficult to assure high prediction accuracy over multiple time points. So the time series modelling approaches are further improved by recurrent neural network (RNN), long short-term memory (LSTM), or dynamic Bayesian networks for perdition accuracy. To improve the prediction accuracy, Wang et al. employed motifs-based dynamic Bayesian networks to estimate the web service's reliability in the near future [18]. The prediction method was further improved by long short-term memory network and self-adaption using data from more nearby time points [19]. As mentioned above, the above researches leveraged historical web service usage data from a single user to construct online prediction rules yet ignored the similarities between the service usage data from multiple users on the same service.

Considering the inner relations in the measurement data from different users, Zheng and Lyu utilized collaborative filtering methods to predict the personalized reliability of web services from the user-side quality [20]. This approach was further refined via neighbourhood integrated matrix factorization [21]. Although this approach considered the spatial correlations of one service among different invokers, they ignored the temporal features of such service over time. The temporal changes of the service quality could not be predicted by the collaborative filter approaches. Moreover, as far as we concerned, there is no existing work

on the time-related service prediction employing invocation history data from multiple users.

On the contrary, many specific system features such as resource usage, environment interactions, and impact of possible faults are all related to event and performance logs. Kubacki and Sosnowski analysed different classes of logs separately and targeted at some specific problems [22]. Event log is used to extract critical event sequences through text processing and classification [23]. Performance log is used to extract the time series of numerical values about system features based on the pulse analysis of operational profiles [24]. In particular, the sequences and periodic groups, characteristic statistical distributions, and identification of data changes with time could all be selected while correlation studies are facilitated to find anomalies or suspicious situations. Based on the measurement logs, Malek built a three-phase failure predictor. A widely used filter method named principal component analysis (PCA) was used to convert a set of correlated features into a set of linearly uncorrelated features using orthogonal transformation. To improve prediction quality, an ensemble of predictors may be used. Having in mind a large number of existing failure prediction algorithms, the most viable solution was heuristically selected after iterative evaluation and refinement [25, 26]. The error log data are also used to model the failure occurrence, similarities between failures, and the relationship between failure events, especially for the web software [27]. The above approaches generally ask for the internal parameters, the operational profiles, or the log files of the software system. The time-domain models depend on the internal parameters or the log files to predict the field behaviour of the program. However, the historical execution and maintenance information is difficult to obtain in the open service-oriented environment.

3. Problem Formulation

The runtime quality of web service can be measured through multiple metrics such as hazard rate, failure probability, and mean time to failure (MTTF). As for the uncertain running environment of web services, steady state or interval measures such as the time interval between successive failures are not suitable for the online quality prediction. Thus, instantaneous measures such as response time and throughput, which are convenient to collect in the runtime, are used for service quality prediction.

The runtime service quality prediction is to predict the quality time series in the future for a period of time according to the reliability data for component services in the past. As shown in Figure 1, the effective time period for the online prediction is Δt_p while Δt_h represents the recent historical time period. A user may call the service after time t, and the purpose of runtime quality prediction is to predict the system quality time series in the period of Δt_p.

Following the quality time series definition in [19], let q_i denote the service quality, either the response time or the throughput value, in the i-th time point. The corresponding runtime quality time series measured during Δt_h is denoted as q_h, which is a vector describing the quality value in n time points:

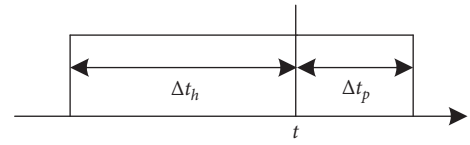

Figure 1: Runtime quality prediction.

$$q_h = (q_1, q_2, q_3, \ldots, q_n). \tag{1}$$

Let q represent a long-term continuously observed quality parameter of the web service. The long-term time series could be divided into t continuous time series slices, each measured by Δt_h. It is obvious that $q = \{q_{h_1}, q_{h_2}, q_{h_3}, \ldots, q_{h_t}\}$, where q_{h_i} denotes the i-th history time series slice of Δt_h. Similarly, let q_p denote the quality time series during Δt_p. The runtime service quality prediction is defined as predicting values of q_p for the next several time points from the given time series q_h:

$$\widehat{q}_p = f(q_h), \tag{2}$$

where $\widehat{q}_p$ denotes the predicted value of quality time series q_p. The function f is also the prediction model.

Since one web service is often invoked by multiple users in the open environment, different users may have similar quality experience from the same service. The runtime service quality prediction problem using different sets of data from multiple users is shown in Figure 2.

Let q_i^j denote the service quality measured from the j-th user in the i-th time point. For the same time period Δt_h, let q_h^j represent the history quality time series from the j-th user measured during this period: $q_h^j = (q_1^j, q_2^j, q_3^j, \ldots, q_n^j)$. The quality of one service measured by m users during Δt_h is defined as a multivariate time series $Q_h = (q_h^1, q_h^2, q_h^3, \ldots, q_h^m)$. Similar as the univariate time series, q^j is the long-term continuously observed quality parameter of one web service from the j-th user and Q is the corresponding long-term time series from multiple users. Let q_p^j denote the quality time series from the j-th user during a period of Δt_p. Q_p represents the multivariate time series of q_p^j. The runtime service quality prediction problem using multiple datasets is formulated as a multivariate time series prediction problem as

$$\begin{cases} \widehat{q}_p^1 = f(q_h^1), \\ \widehat{q}_p^2 = f(q_h^2), \\ \vdots \\ \widehat{q}_p^m = f(q_h^m), \end{cases} \tag{3}$$

where $\hat{q}_p^j$ denotes the predicted value of each quality time series q_p^j. For each group of historical data from different users, a unified function f is built for different time series forecasting at the same time. So, the main work of this paper is to find an appropriate method to solve f.

4. Online Quality Prediction Method

In this section, an online service quality prediction method is proposed. The core idea is a multivariate time series model

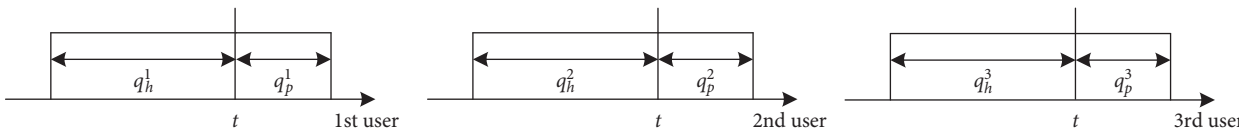

Figure 2: Runtime quality prediction of multiple users.

which employs the LSTM network and multiple quality time series data.

4.1. Runtime Quality Time Series Statistical Analysis. Before selecting an appropriate online prediction method, the features of the runtime quality time series should be analysed. For example, stationary time series forecasting could be handled well by autoregressive-moving-average (ARMA) models. Nevertheless, nonlinear patterns in the time series data could not be handled by statistical models. So, in the beginning, the stationarity of each service quality time series is checked through the autocorrelation and the partial-autocorrelation functions. Given one long-term quality time series $q = (q_1, q_2, q_3, \ldots, q_N)$, the autocorrelation is computed as

$$R_{qq}(\tau) = \frac{1}{(N-\tau)\mathrm{std}(q)^2} \sum_{i=1}^{N-\tau} (q_i - E(q))(q_{i+\tau} - E(q)), \tag{4}$$

where $E(q)$ and $\mathrm{std}(q)^2$ are the mean and variance of q. The partial autocorrelation is defined as the correlation between the residuals resulting from the linear regression of q with respect to a set of controlling random variables. An example of the autocorrelation and partial-autocorrelation values is shown in Figure 3. The values in Figure 3 are computed from two quality time series from two users invoking the same service.

It is observed that the quality time series from different users exhibit different statistical characteristics, even for the same service. For the first user, the autocorrelation of both the response time and the throughput data converges rapidly to remain below the significance range, under the default confidence level 0.95. This is the indication of a stationary series. However, the corresponding value of the time series from the second user is decreasing very slowly and remains well above the significance range. This is an indicative of a nonstationary series. As for the partial-autocorrelation function, the values fluctuate dramatically, which prevents from selecting appropriate orders for the ARMA models. The autocorrelation and partial-autocorrelation tests are conducted on a group of time series data from different users for the same web service. The results show that some quality time series are individually stationary while the others are nonstationary. To get a confirmatory evidence about the time series stationarity, the augmented Dickey–Fuller (ADF) test is used to check whether the expectations of stationarity are met or not. The results are shown in the experiments section.

In order to check whether multiple quality time series are jointly wide-sense stationary, the correlation between these time series are evaluated. Given two long-term quality time series q^j and q^k from two different users, the correlation between them is computed using the Pearson correlation coefficient:

$$\rho_{j,k} = \mathrm{corr}(q^j, q^k) = \frac{1}{(N-1)\mathrm{std}(q^j)\mathrm{std}(q^k)} \cdot \sum_{i=1}^{n} (q_i^j - E(q^j))(q_i^k - E(q^k)), \tag{5}$$

where $E(q^j)$ and $E(q^k)$ are the mean values of q^j and q^k and $\mathrm{std}(q^j)$ and $\mathrm{std}(q^k)$ are the corresponding standard deviations. A group of quality data from 10 users is used for illustration, and the correlation value between each pair of quality time series is shown in Figure 4.

The data in Figure 4 indicate that there is no obvious linear relationship between these quality time series, although very few time series are highly correlated to each other. In case there is time delay between these time series, the lagged cross-correlation between these time series is also checked as

$$\rho_{j,k}(\tau) = \frac{1}{(N-\tau)\mathrm{std}(q^j)\mathrm{std}(q^k)} \sum_{i=1}^{N-\tau} (q_i^j - E(q^j))(q_{i+\tau}^k - E(q^k)). \tag{6}$$

The cross-correlation values among 4 response time series and 4 throughput time series are drawn as an example in Figure 5. Data in Figure 5 show that no obvious linear patterns exist in the cross-correlation results, while some pairs of time series may be linearly related to each other. It indicates that there may not be obvious linear relationship between these time series. The above analysis results could not guarantee the jointly wide-sense stationarity between two time series, not to mention the stationarity of each web service quality time series.

While there is no obvious linear relationship between these time series, the feasibility of building a unified time series model over multiple time series is evaluated through nonlinear relationship analysis. The time series relationship could be checked by the cointegration between two time series. The augmented Engle–Granger two-step test is used to check the cointegration, and the results are shown in the experiments part. Test results indicate that there are similarities between multiple time series.

Since time series are similar to but not linear correlated with each other, nonlinear patterns and nonlinear relations should be handled during the quality time series forecasting process. ARMA-related models are not suited for this kind of time series. There are multiple ways to build nonstationary time series forecasting methods such as ARIMA and RNN.

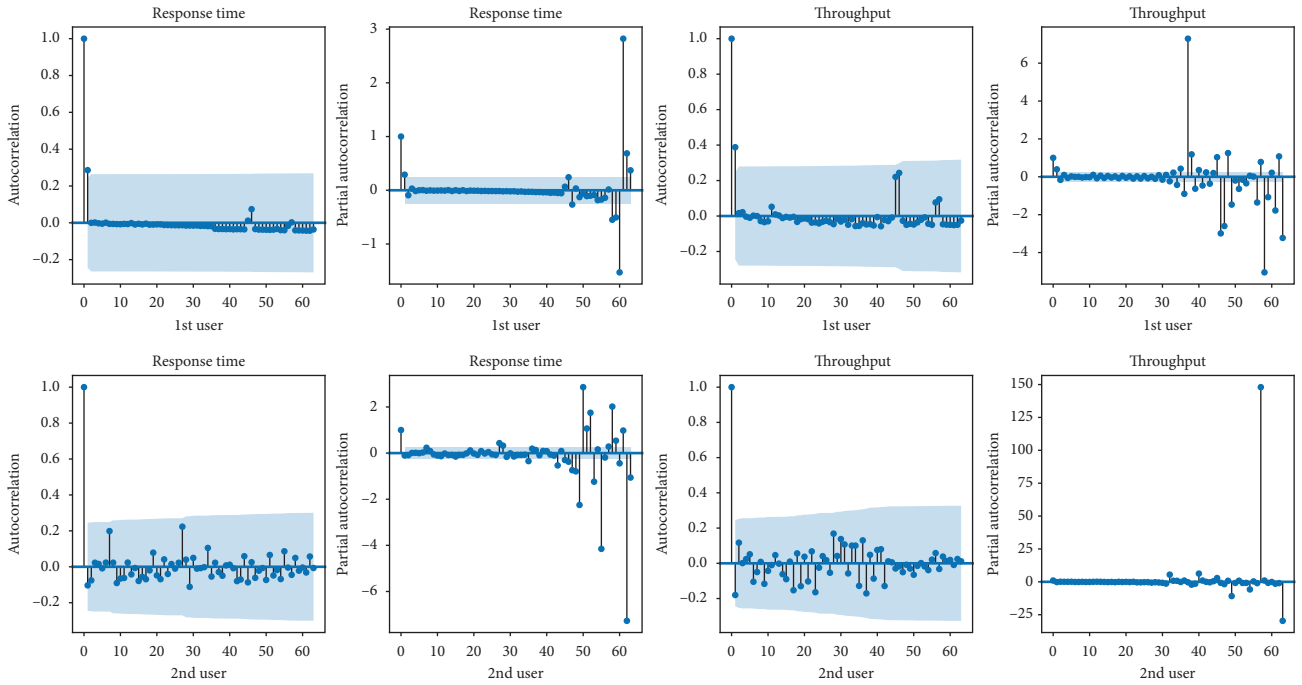

Figure 3: An example of the autocorrelation and partial-autocorrelation test for 4 time series.

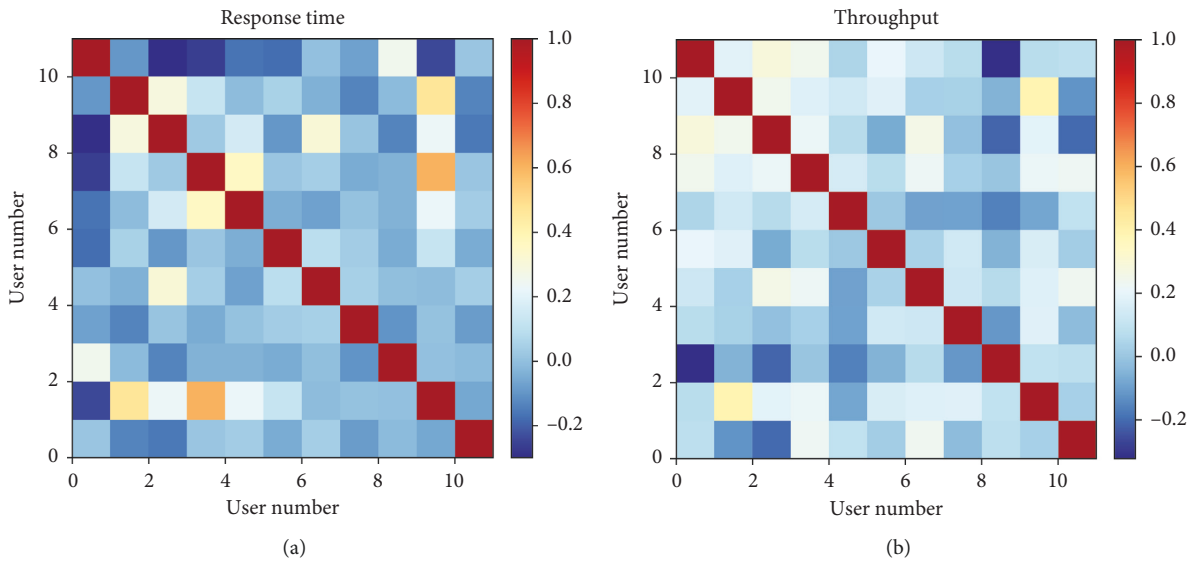

Figure 4: An example of the correlation values for 10 time series.

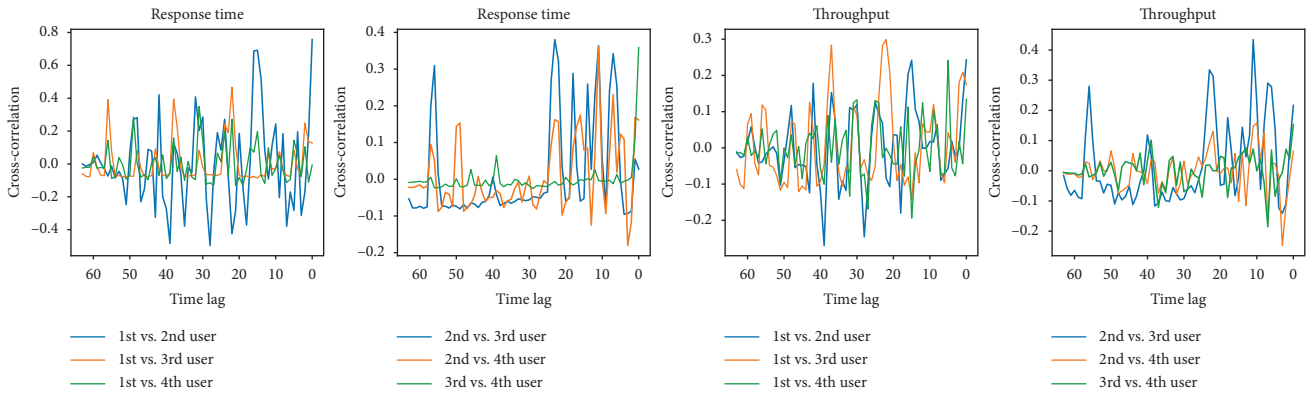

Figure 5: An example of the cross-correlation values for 4 time series.

As new-emerging technology, RNNs have advantages in dealing with predicting nonlinear and complex patterns from historical data. However, a traditional RNN suffers from the vanishing gradient problem and the long short-term memory (LSTM) network is proposed for the time series forecasting scenario [28]. So, LSTM prediction methods are borrowed to build time series forecasting models in the following subsection.

4.2. Multivariate Long Short-Term Memory Network. The framework of the LSTM network is shown in Figure 6, where q_{i-1}^j denotes the historical service quality measured in the historical time point and $\widehat{q}_i^j$ denotes the quality data predicted for the i-th time point. LSTMs have the chain-like structure, but the repeating module has a different structure from RNN. The key to LSTMs is the cell state, which is shown in Figure 7. Instead of having a single neural network layer, there are four layers interacting in a very special way. The values of f_i^j, I_i^j, g_i^j, and o_i^j, denote the different network layers during the prediction in the i-th time point, and c_i^j denotes the cell state. The value of $\widehat{q}_i^j$ is concluded from the values of q_{i-1}^j and $\widehat{q}_{i-1}^j$ through the combination of the four layers.

Nonlinear models could be built using LSTM to handle multiple dimension data in nonlinear way. For the multivariate time series, the multivariate LSTM model is constructed as follows. For a certain time point i, let $\mathbf{q}_i$ denote the vector describing the web service quality value from multiple users:

$$\mathbf{q}_i = \left(q_i^1, q_i^2, q_i^3, \ldots, q_i^m\right)^{\mathrm{T}}, \tag{7}$$

and $\widehat{\mathbf{q}}_i$ denote the corresponding vector containing the prediction quality value. For simplicity, the intermediate results from the four layers are all denoted by the vector-matrix form as $\mathbf{f}_i$, $\mathbf{I}_i$, $\mathbf{g}_i$, $\mathbf{o}_i$, and so on.

In each time point, the information from the historical quality data is input along with the prediction results from the last time point:

$$\mathbf{f}_i = \sigma\left(\mathbf{W}_{qf}\mathbf{q}_{i-1} + \mathbf{W}_{\widehat{q}f}\widehat{\mathbf{q}}_{i-1} + \mathbf{b}_f\right). \tag{8}$$

The next step is to decide what new information is going to store in the cell state. Firstly, a sigmoid layer, called the input gate layer, decides which values should be updated. Next, a tan$h$ layer creates a vector of new candidate values, which could be added to the state:

$$\begin{aligned}\mathbf{I}_i &= \sigma\left(\mathbf{W}_{qI}\mathbf{q}_{i-1} + \mathbf{W}_{\widehat{q}I}\widehat{\mathbf{q}}_{i-1} + \mathbf{b}_I\right),\\ \mathbf{g}_i &= \tanh\left(\mathbf{W}_{qg}\mathbf{q}_{i-1} + \mathbf{W}_{\widehat{q}g}\widehat{\mathbf{q}}_{i-1} + \mathbf{b}_g\right).\end{aligned} \tag{9}$$

In the next step, the above two states are combined to create an update to the state:

$$\mathbf{c}_i = \mathbf{f}_i\mathbf{c}_{i-1} + \mathbf{I}_i\mathbf{g}_i. \tag{10}$$

Finally, the output is generated based on the cell state. Firstly, a sigmoid layer decides which part of the cell state is going to output. Then, the cell state is put through a tan$h$ layer and multiplied by the output of the sigmoid gate:

$$\begin{aligned}\mathbf{o}_i &= \sigma\left(\mathbf{W}_{qo}\mathbf{q}_{i-1} + \mathbf{W}_{\widehat{q}o}\widehat{\mathbf{q}}_{i-1} + \mathbf{b}_o\right),\\ \widehat{\mathbf{q}}_i &= \mathbf{o}_i\tanh\left(\mathbf{c}_i\right).\end{aligned} \tag{11}$$

The loss L between the predicted value and the observed value over an entire sequence with the length of n is defined as

$$L = \frac{1}{|n|}\sum_{i\in n}\left\|\widehat{\mathbf{q}}_i - \mathbf{q}_i\right\|^2. \tag{12}$$

The weight matrices $\mathbf{W}$ and the bias vectors $\mathbf{b}$ are adjusted through historical data training to minimize the loss L through the gradient descent. The training process continues until the loss L converges. The detailed calculation process could refer to [29] and is out of the scope of this paper.

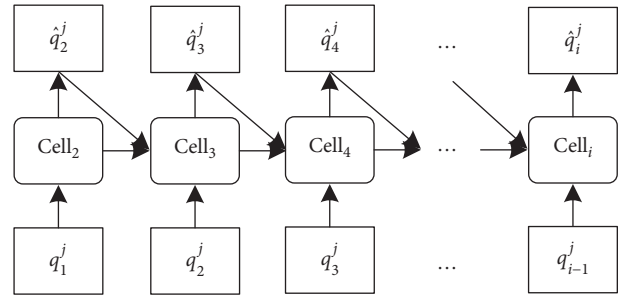

Figure 6: An LSTM network.

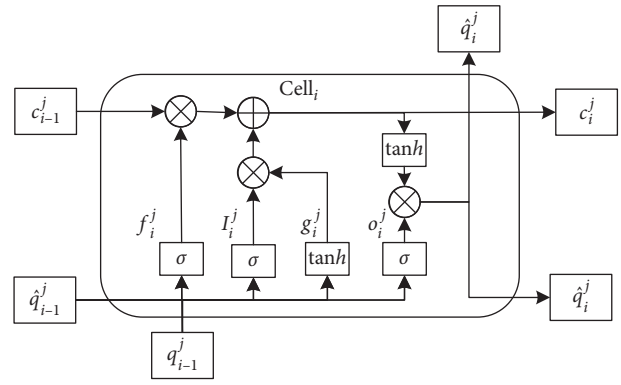

Figure 7: An example of the LSTM network cell.

4.3. Online Service Quality Prediction Algorithm. Based on the multivariate LSTM model, an online prediction algorithm is proposed in this section for runtime service quality forecasting.

4.3.1. Multivariate Time Series Selection and Alignment. Since the service quality time series may be affected by the user's network or machine delay, a set of time series with similar patterns should be selected first. To reduce the randomness among the service quality from different users, the odd service quality data from certain users should be eliminated from the dataset. The distance between two sets of time series data is used for the time series similarity evaluation.

Due to the potential data frame mismatch, the similarity between two time series could not be evaluated by the traditional Euclidean distance effectively. Instead, dynamic time warping (DTW) is employed in this section [30]. First of all, the potential time delay between two long-term time series q^j and q^k is evaluated through the time-lag cross-correlation coefficient:

$$\left(q^j * q^k\right)(\tau) = \sum_{i=1}^{N-\tau} q_i^j q_{i+\tau}^k. \tag{13}$$

The maximum value of this function indicates the point in time where the two time series are best aligned. So, the optimal time delay is determined by the argument of the maximum cross-correlation:

$$\tau_{\text{optimal}} = \text{argmax}\left(q^j * q^k\right). \tag{14}$$

Secondly, let q_h^j and q_h^k denote two time series with the length of n_j and n_k, respectively, while q_i^j and q_i^k represent the i-th quality values in the series data. A warping path w between the nodes in q^j and q^k is defined as a sequence $w = (w_1, w_2, w_3, \ldots, w_{|w|})$, where $w_y = (i_y^j, i_y^k)$, $1 \le y \le |w|$. The values of i_y^j and i_y^k denote the correspond time points or index of the node in the time series q_h^j and q_h^k, satisfying the following conditions:

(i) Boundary condition: $w_1 = (1, 1)$ and $w_{|w|} = (n_j, n_k)$

(ii) Monotonicity and continuity condition: for any two values $w_{y+1} = (i_{y+1}^j, i_{y+1}^k) \neq w_y = (i_y^j, i_y^k)$, it is satisfied that $i_y^j \le i_{y+1}^j \le i_y^j + 1$ and $i_y^k \le i_{y+1}^k \le i_y^k + 1$

The distance $D(i_y^j, i_y^k)$ of the path w_y is defined as

$$D\left(i_y^j, i_y^k\right) = \left|q_{i_y^j}^j - q_{i_y^k}^k\right| + \min\left\{\left(i_y^j - 1, i_y^k\right), D\left(i_y^j, i_y^k - 1\right), D\left(i_y^j - 1, i_y^k - 1\right)\right\}. \tag{15}$$

The DTW distance is the sum of the pointwise distances along the optimal path w^*, with the smallest cost $c(w)$:

$$w^* = \text{argmin}\,(c(w)), \tag{16}$$

where $c(w) = \sum_{y=1}^{|w|} D(i_y^j, i_y^k)$.

It is noted that the DTW distance is the value of $c(w^*)$. A smaller value of DTW distance indicates a higher similarity between two time series. The implementation of DTW can be optimized through different ways such as FastDTW [31] and will not be discussed thoroughly in this paper. Under the similarity analysis using DTW, the group of time series is selected using Algorithm 1.

4.3.2. Online Prediction Algorithm. Based on the multivariate time series analysis methods, an online service quality prediction algorithm is constructed using Algorithm 2.

5. Experimental Evaluation and Discussion

In this section, experimental evaluation is presented for the proposed prediction method. The WS-DREAM repository (https://github.com/wsdream/wsdream-dataset), which contains two open web service quality datasets, is used in this section. The datasets describe QoS measurements collected from thousands of services in the real world. The application of these services varies from weather forecasting, city service, bank service, e-mail, authentication, and so on. The location of the service provider also distributes all around the world, from Asia to Europe. The information about the IP address, WSDL address, and service provider of each service could refer to dataset#1 in WS-DREAM [32]. The QoS measurement data are collected by invoking the services one by one from different locations at different time. The IP address and location of the invokers are also included in dataset#1. Dataset2# provides the real-world QoS measurements from 142 users on thousands of web services over 64 consecutive time slices at 15-minute interval [33]. The continuous measurement data could be used to build the time series model, so the measurements in dataset#2 are mainly used in the following experiments.

According to this dataset, two types of quality measurements are employed as response time and throughput. The throughput is represented as the data size successfully transmitted within a unit time from the web service. Different from the synthetic data, the quality data collected in the real network environment may exhibit nonstationary or even unpredictable statistical characteristics, which could not be handled by traditional models. Furthermore, the diversity and uncertainty in the data among different services may not be handled by the univariate models. So, it is reasonable and feasible to apply multivariate LSTM to this dataset. The efficiency of multivariate LSTM in this dataset also proves the efficiency of the method in the nonstationary time series prediction.

5.1. Time Series Statistical Analysis. To get a confirmatory evidence about the time series stationarity, the augmented Dickey–Fuller (ADF) test is used to check whether the expectations of stationarity are met. The ADF test is a kind of unit root test, for which the null hypothesis is that the time series possesses a unit root and hence is not stationary. If the test statistics p value is less than a critical value, then the null hypothesis that there is a unit root is rejected. Hence, the time series used for the test is stationary. Otherwise, it fails to reject the null hypothesis, which suggests that the series data are nonstationary. A group of example time series data from 20 web services is chosen for illustration in this experiment for the stationary test. For each web service, the response time or throughput time series is collected from 142 different users. The ADF test is conducted on a total of 5680 time series, and it appears that nearly 82.11% of them are stationary time series, under the confidence level 95%. The ratio of stationary time series over each service is collected and shown in Table 1. According to the stationary test results, while most of the time series are stationary, there are a few nonstationary time series in the dataset. So, the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests are also conducted on nonstationary time series to check whether they are stationary around a deterministic trend against a

Input: a group of long-term time series data $Q = (q^1, q^2, q^3, \ldots, q^m)$, the maximal size z of the subset, and the similarity threshold ε
Output: a subset of Q, in which any time series is highly similar to each other
(1) Initialize distance matrix $\mathbf{DM} = 0_{m\times m}$.
(2) For each node (j, k) in $\mathbf{DM}$:
(i) If $j \neq k$,
(a) First, calculate the time-lag cross-correlation coefficient of any two time series q^j and q^k to get the optimal time delay between them, and align the two time series according to the optimal time delay.
(b) Then, calculate the values of $\mathbf{DM}(j,k) = \mathbf{DM}(k,j) = c(w^*)$, where w^* is the optimal warping path between the aligned q^j and q^k.
(ii) Else, $\mathbf{DM}(j,j) = \infty$.
(3) After calculating the distance matrix $\mathbf{DM}$, calculate the similarity of a time series to the rest of other time series data as $s(q^j) = \sum_{k=1\%\&k\neq j}^{m} \mathbf{DM}(j,k)$, $1 \le j \le m$.
(4) Sort the time series according to $s(q^j)$ in ascending order: $q^j < q^k$ if and only if $s(q^j) < s(q^k) < \varepsilon$.
(5) Select a subset of Q according to the following rules. For the sorted time series $Q^* = (q^{1*}, q^{2*}, q^{3*}, \ldots, q^{m*})$, choose the first z number of time series as the final subset.

Algorithm 1: Time series selection and alignment.

Input: a group of long-term quality time series data $Q = (q^1, q^2, q^3, \ldots, q^m)$ and the number of time points n_p for forecasting in the near future.
Output: a group of quality prediction results $\widehat{\mathbf{q}}_i\,(i < n_p)$.
(1) Data selection: employ Algorithm 1 to select a subset of time series from Q, in which any time series is highly similar to each other. In the selection process, align the selected time series according to the optimal time delay.
(2) Data preparation: divide each long-term time series q^j into multiple time series q_h^j, each measured during Δt_h with n time points. The moving time window for the division is a single time point. Each time series is related with an observed time series data q_p^j for training. The data in q_p^j are measured during Δt_p, with n_p time points. Thus, the whole dataset Q is divided into multiple multivariate time series $Q_h = \{q_h^1, q_h^2, q_h^3, \ldots, q_h^m\}$, and each multivariate time series is related with an observed time series $Q_p = \{q_p^1, q_p^2, q_p^3, \ldots, q_p^m\}$ for training.
(3) Model training and evaluation: train the multivariate LSTM network and compute the weight matrices $\mathbf{W}$ the bias vectors $\mathbf{b}$ using the input training data set. Evaluate the loss L on the evaluation data set to adjust the model parameters.
(4) Model prediction: during the n_p time slices, take the historical quality time series to predict the value in the near feature. Update the model with the new input prediction results.
(5) Model update: after n_p time slices elapses, update the historical training set with the new observed result and retrain the prediction model.

Algorithm 2: Online service quality prediction.

Table 1: The ratio of stationary time series for each web service.

Service no.	Response time series (%)	Throughput time series (%)	Service no.	Response time series (%)	Throughput time series (%)
1	92.31	95.73	11	90.60	79.49
2	90.60	92.31	12	76.92	70.94
3	87.18	88.89	13	78.63	47.01
4	84.62	91.45	14	69.23	93.16
5	16.24	88.03	15	92.31	96.58
6	88.89	84.62	16	92.31	94.02
7	86.32	85.47	17	95.73	86.32
8	86.32	88.03	18	91.45	90.60
9	94.02	81.20	19	91.45	91.45
10	78.63	83.76	20	92.31	91.45

unit root. The results show that none of these time series passes the KPSS tests.

To check the feasibility of the multivariate forecasting model, the cointegration test is taken on the series data from 20 services. For each service, a group of time series data is randomly chosen from different users and the test is carried out between any pair of time series in each group of data. Similar as the ADF test, if the resulting p value is less than a critical value, then the null hypothesis that there is no cointegrating relationship between the two time series is rejected and the two time series are cointegrated. Otherwise, it fails to reject the null hypothesis, which suggests that the two time series are noncointegrated. The ratio of cointegrated time series in for each service is show in Table 2.

Table 2: The ratio of cointegrated time series for each web service.

Service no.	Response time series (%)	Throughput time series (%)	Service no.	Response time series (%)	Throughput time series (%)
1	90.00	91.11	11	88.89	90.00
2	80.00	90.00	12	85.56	65.56
3	88.89	100.00	13	90.00	78.89
4	100.00	100.00	14	77.78	100.00
5	56.67	94.44	15	100.00	86.67
6	88.89	97.78	16	85.56	93.33
7	98.89	86.67	17	85.56	88.89
8	86.67	96.67	18	76.67	68.89
9	90.00	71.11	19	55.56	96.67
10	78.89	70.00	20	71.11	78.89

The results show that most of the time series in the same group are cointegrated with each other. Those that are not cointegrated may be affected by the time delay between a pair of web service or by the random error in the data.

To get the optimal time delay between any two time series, the time-lag cross-correlation coefficient is evaluated. Examples of the cross-correlation coefficients between two time series from different users for the same service are shown in Figure 8. From the results in Figure 8, when the time delay between each two time series is close to 0, the two time series achieve the highest cross-correlation value. No obvious time delay is observed from the time series data in this dataset. The results indicate that the time series data from different users are collected during a close period of time.

5.2. Multivariate Time Series Forecasting

5.2.1. Multivariate LSTM Model Construction. After the above data statistical analysis, the forecasting model is built on the multivariate LSTM network. An example of the data prediction using the above multivariate LSTM model is shown in Figure 9. The time series data from multiple users over a single service can be predicted at the same time, and this figure shows the forecasting results for four users. Except for the first few time points, the predicted time series and the observed time series generally overlap. Data in Figure 9 show that it is feasible to employ multiple time series for future prediction and the multivariate forecasting model exhibits a relatively high prediction accuracy.

The prediction accuracy of the LSTM model can be evaluated by two widely adopted indicators: mean absolute error (MAE) and root mean square error (RMSE) between the predicted value and the observed value. The value of MAE and RMSE is computed as

$$\begin{aligned} \text{MAE} &= \frac{\sum_{i=1}^{n} \left| \widehat{q}_i - q_i \right|}{n}, \\ \text{RMSE} &= \sqrt{\frac{\sum_{i=1}^{n} \left(\widehat{q}_i - q_i \right)^2}{n}}, \end{aligned} \tag{17}$$

where n denotes the number of predictions, q_i is the original observation value, and $\widehat{q}_i$ is the predicted value. Smaller values of MAE and RMSE indicate better prediction accuracy [19].

As a special type of neural network, LSTM needs to be trained in advance. Different hypoparameters chosen in the training process may result in different prediction accuracy. Therefore, to choose an appropriate group of hypoparameters, experiments are first conducted to compare the prediction accuracy using different groups of parameters. The process of choosing certain parameters, which have significant impact on the result accuracy, is presented as follows.

First of all, the optimal number of variants in the multivariate forecasting model is determined. Considering statistics error, time series data from 5 different services are used for comparison. For each service, the number of time series used to train the LSTM model varies from 1 to 10. The corresponding MAE and RMSE of the forecasting model using different number of variants for different service are show in Figure 10.

Data in Figure 10 show that as the number of time series m increases, the MAE and RMSE value of the LSTM model decreases dramatically at first. This is because multiple sets of time series could help to reduce the side effects of errors of each time series in the forecasting model. Then, the error decreasing trend declines gradually. For some web services (Service No. 4 and Service No. 5), the MAE and RMSE values maintain at a relatively stable level when the number of variants reaches 7. For the others (Service No. 2), the MAE and RMSE even increase when the number of variants continues to increase. It is because that too many numbers of variants may also cause over-converge and the accuracy of the forecasting model decreases in other datasets. Therefore, there is no need to increase the number of variants continuously. Considering to achieve the balance between the algorithm efficiency and accuracy, the number of variants used to train the multivariate LSTM model is set to be 7.

After that, the optimal training steps for the LSTM model are chosen from experiments. The MAE and RMSE of the forecasting model using 500 to 3000 training steps are evaluated. Time series data from 5 services are also used in this experiment, and the results are show in Figure 11.

Data in Figure 11 show that the prediction error of the LSTM model decreases with the increase of the training steps. More training steps help to reduce the loss value during the training process, and the prediction accuracy increases accordingly. However, after 2000 training steps, the prediction error generally stays stable, even the training steps continues to grow. The results indicate that there is no need to increase the training steps without limits and an appropriate value also should be chosen for the algorithm

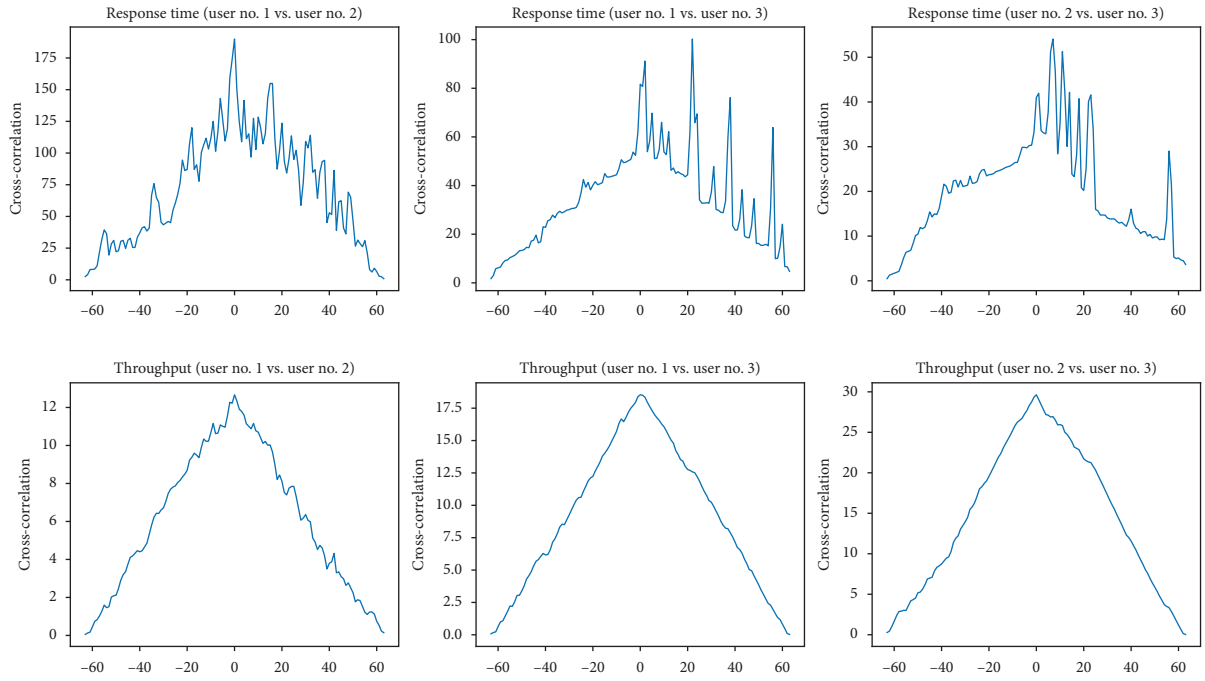

Figure 8: Cross-correlation between two time series of the same service.

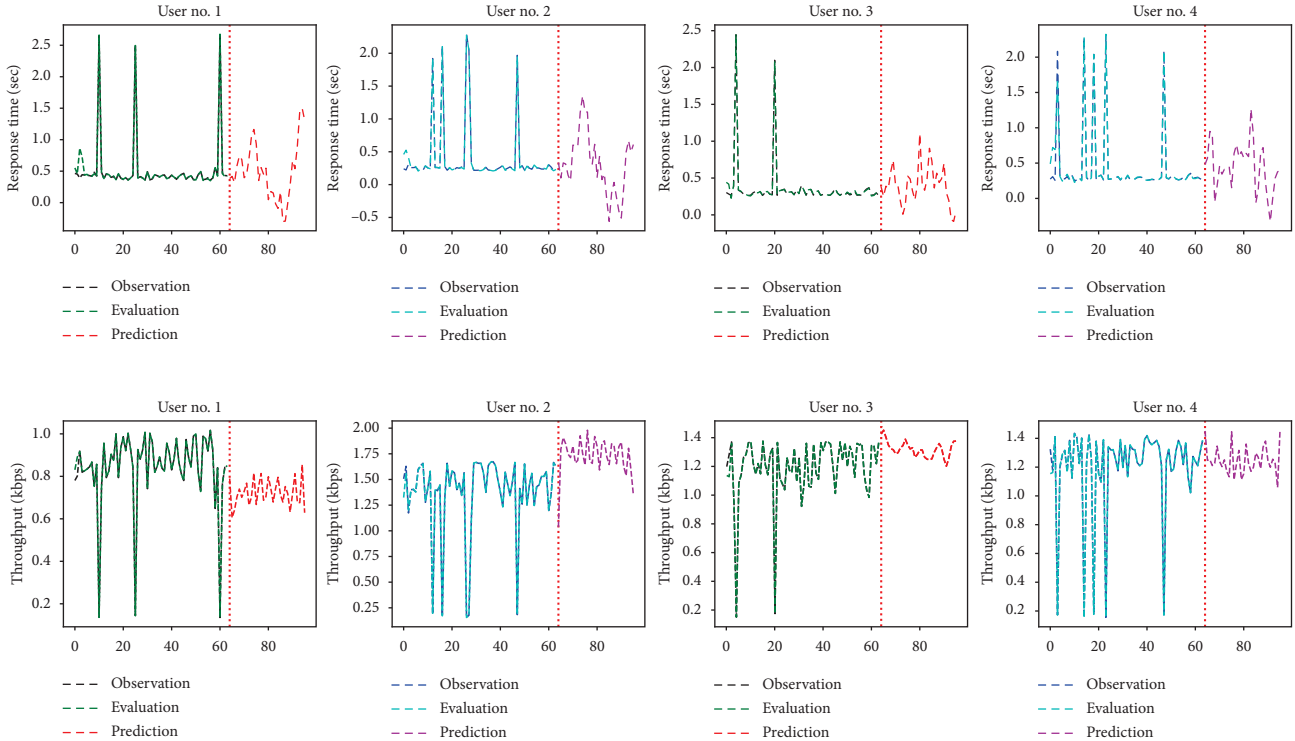

Figure 9: The prediction results from the multivariate LSTM model.

efficiency. According to the test results, a multivariate LSTM model with 7 time series and 2000 training steps is built.

5.2.2. Multivariate Forecasting Model Comparison. After building the model, experiments are conducted to compare the prediction accuracy between this model and the traditional models. In the first step, the multivariate model is compared with a univariate LSTM model. The web service usage dataset from a single service is used in this experiment. In each test case, a different subset of usage data from the same web service is used. For the multivariate LSTM model,

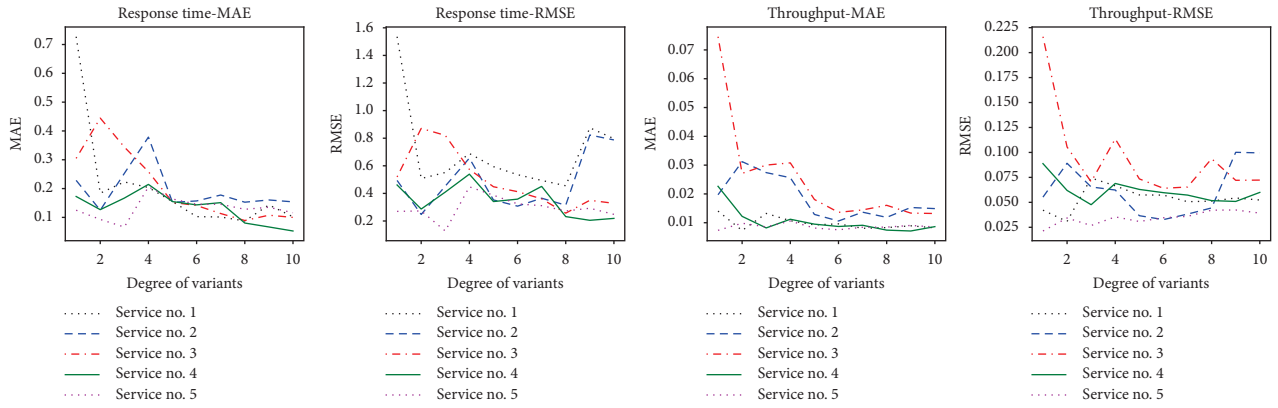

Figure 10: The prediction accuracy of the multivariate LSTM model using different numbers of time series.

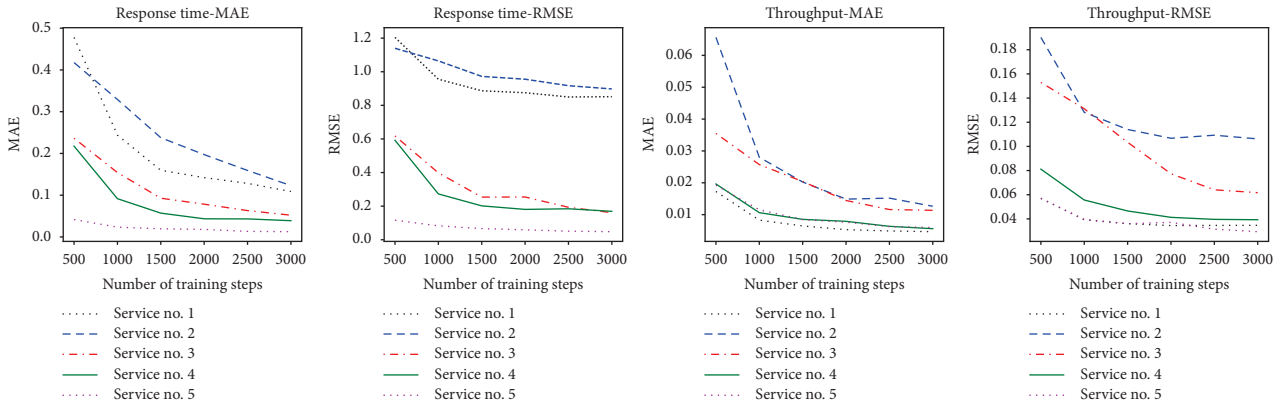

Figure 11: The prediction accuracy of the multivariate LSTM model using different numbers of training steps.

a group of data from 7 users is taken use of. For the univariate model, the web service usage data from a single user are employed. The MAE and RMSE results of 12 groups of subset data prediction are shown in Figure 12, where M-LSTM stands for the results from the multivariate LSTM model while U-LSTM represents the results of the univariate model. Comparisons between M-LSTM and U-LSTM show that the MAE value of the multivariate LSTM model is generally less than 0.1, both for response time prediction and for throughput prediction. It is much lower than the MAE value of the univariate LSTM model, which is generally around 0.3 in each test case. The similar conclusion could be drawn for the value of RMSE. It indicates that the prediction error is largely reduced by introducing multiple variants in the prediction model.

Then, the nonlinear LSTM model is also compared with traditional linear ARMA model. A univariate ARMA model with one time series and a multivariate ARMA model with 7 time series are built to compare with our model. The results of MAE and RMSE are also shown in Figure 12. U-ARMA stands for the results from the univariate ARMA model while M-ARMA represents the corresponding results of the multivariate ARMA model. It is shown that, for response time prediction, the errors of U-ARMA are generally less than that of M-ARMA. For the throughput value prediction, the MAE and RMSE values of M-ARMA are generally lower in most test cases. Taking the multivariate LSTM model into consideration, the MAE and RMSE values of M-ARMA are higher than those of M-LSTM in most test cases, indicating worse prediction accuracy. The univariate ARMA model performs well in some test cases. For example, the MAE value of U-ARMA in the first test case of response time forecasting is lower than that of the M-LSTM. However, the performance of the ARMA models is not quite stable. The MAE values of the U-ARMA model fluctuate violently among different test cases from less than 0.1 to around 1. On the contrary, the MAE of M-LSTM stays below 0.1 in most test cases. The unstable performance of the ARMA model could be blamed to the lack of statistical stability of some time series. While the ARMA model is built upon the statistical features of each time series, different statistical features may lead to different prediction performance. Since the multivariate LSTM model does not ask for the time series stationarity, the performance is more stable for the real data collected in the open environment.

After that, the comparison among different kinds of models is conducted using the time series data from different web services. Time series data from 12 services are used in this experiment. For each service, a group of time series data from 7 users is used for the comparison. The prediction results from 4 types of forecasting models are calculated, and the corresponding MAE and RMSE are shown in Figure 13. A similar phenomenon could be observed from the

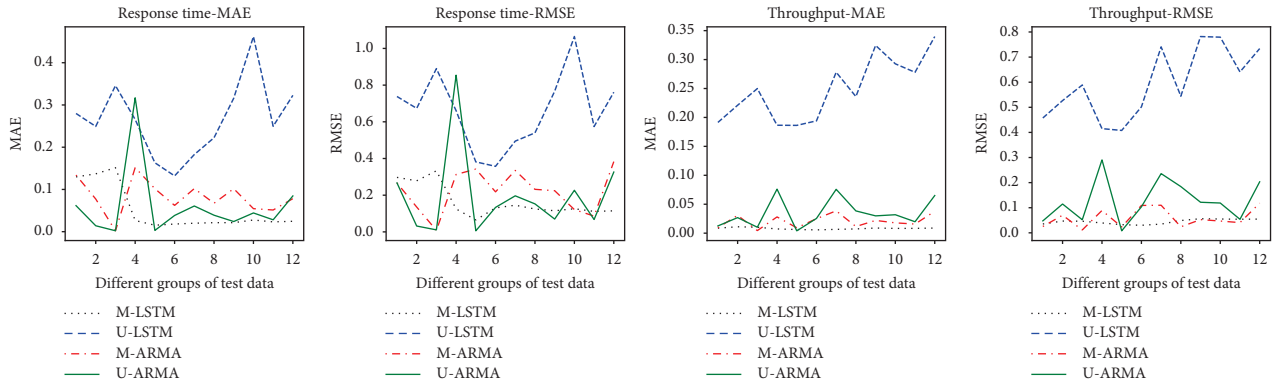

Figure 12: The prediction accuracy comparison between the multivariate LSTM model and the other models.

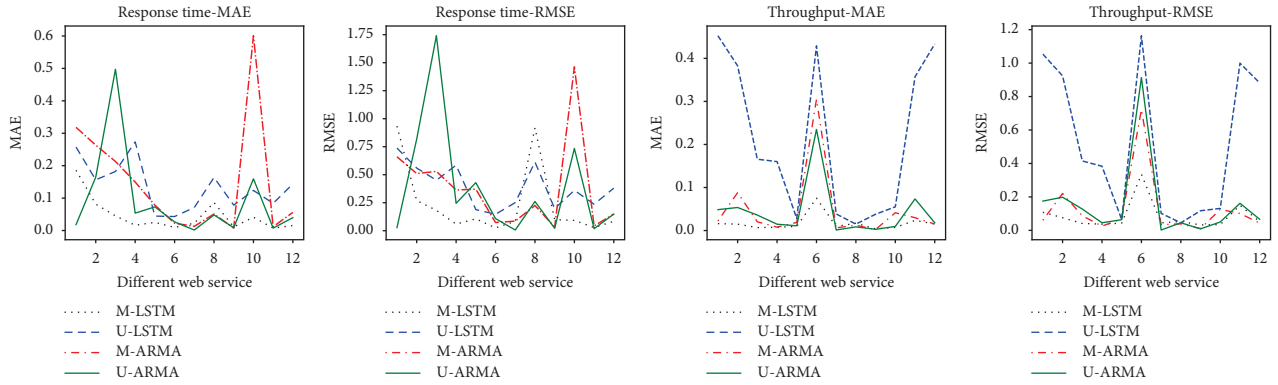

Figure 13: The comparison between the multivariate LSTM and the other models using different web services.

Figure 13 that the prediction error, both MAE and RMSE, of M-LSTM is less than that of U-LSTM in each test case. The prediction accuracy of M-ARMA and U-ARMA outperforms each other in different test cases. In very few test cases, the prediction accuracy of U-ARMA outperforms the other three models. However, the performance of ARMA models is not so stable since the MAE and RMSE values change from less than 0.1 to more than 1. In most test cases, the MAE and RMSE values of M-LSTM stay constantly below 0.1, which is less than the value of the other three models. The performance of the multivariate LSTM model is more stable among different services.

Generally speaking, multivariate prediction models outperform the univariate models in both MAE and RMSE. In most test cases, the multivariate LSTM model achieves better and more stable performance in quality prediction. The comparison results suggest that employing multivariate time series analysis methods is more efficient in the service quality online prediction problem.

6. Threats to Validity

A set of potential internal and external threats to the validity of the presented approach is discussed in this section.

6.1. Internal Validity. The quality dataset used in our study is garnered from a public data repository, which is highly referred in many researches on web service quality prediction. It leads us to believe that this dataset is generally sound and correct. So, the main threat remains to the training data selection inside this dataset. Although the whole group of quality data on the same service could be chosen easily, there may be some odd time series inside each group. The odd data are introduced by the unpredictable network delay, the sudden application error, and etc. It is not related to the other data inside the same group and may affect the model prediction accuracy during the training process. In this paper, the odd time series is dropped by analysing the similarity and the distance between any pair of time series. However, this preliminary solution could not guarantee the elimination of all the odd data. In very few test cases, the prediction accuracy of the multivariate LSTM model is affected by them according to the experimental results. So, a more precise similarity analysis model such as PCA should be studied for the time series relationship analysis. Moreover, LSTM could be combined with other kinds of network, which are more suitable for relationship analysis instead of time series. We hope that the joint neural network could handle the odd data in the training process, by decreasing the coefficient of the unrelated time series to a very small extent through training.

After that, a related thread concerns to the degree of the selected time series. Considering the algorithm efficiency and accuracy, the optimization idea is employed to choose the degree after several times of experiments. The optimization

objective and constraint models to this problem could be studied in the future, in order to choose the degree in a wise manner.

6.2. External Validity. This validity concerns about the generalization of results obtained from the experimental analysis. Due to the nature of LSTM, different training data may result in varying performance of the presented approach. The findings of the presented study need to be considered within the domain of web service quality prediction only. One potential threat concerns the representativeness of the web services used in the experiments. The web services in the dataset are collected from different domains such as weather forecasting, city service, bank service, e-mail, and authentication. However, the quality data are only collected by some specific invokers. While it is impossible to collect the quality data from every real web service all over the world, the usage of this dataset already proves that the proposed approach could be used for most kinds of general web services. It is also reasonable to suppose that it is applicable to employ the approach to other web service quality data repositories collected by other researchers. The application results could be checked and verified in the next work.

Another potential threat concerns the representativeness of the web service quality indicators. Response time and throughput, which are easy to collect, are used as the quality indicators in this paper. The time series approach could be extended to the similar quality measurement prediction. However, there are some other kinds of system quality measurement, e.g., resource usage, failure times, and reliability. It should be discussed whether the features of these measures follow the same pattern as the response time or throughput. The application of the approach to these measurements should also be verified, in the future work.

7. Conclusions and Future Work

Online quality prediction not only helps to select appropriate services but also helps to identify the quality degradation in preventive maintenance. Since the online service quality prediction is essential to preventive maintenance, much attention has been paid to service prediction. In the traditional case, the historical quality experience from one user is employed and univariate time series models are built to obtain the near future time series. The similarities in the service usage data from multiple users invoking the same service are ignored. To improve the service quality prediction accuracy, an online quality prediction method is built considering multiple user invocation process. A multivariate time series is used to characterize the multiple invocation process, and the time series is estimated using a multivariate LSTM model. Experimental results showed that multivariate LSTM method outperformed the univariate models and the ARMA models in both MAE and RMSE. Besides, the multivariate LSTM model achieved the most stable performance in most test cases. The comparison results proved the efficiency of our method.

According to the experimental evaluation results, the prediction accuracy of the multivariate LSTM model is worse than the ARMA models, in very few test cases. So, the next step of work involves analysing the time series statistical characteristics in detail and improving the efficiency of the multivariate prediction algorithm accordingly.

If the measurement log of a web service could be obtained, there will be extensive other system features for quality analysis. The prediction method could also be improved by introducing PCA analysis or heuristic algorithms [3], and the prediction accuracy could be largely improved. While it is difficult to obtain the measurement log through the network, other information might be collected such as maintenance history. In the next step of work, the method to collect other system features could be considered and the quality prediction method should be extended considering these features.

Data Availability

The web service quality data used to support the findings of this study have been deposited in the GitHub WS-DREAM repository. The details of the dataset are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was financially supported by the Basic Research and Frontier Exploration (Natural Science Foundation) Projects in Chongqing (Grant no. cstc2019jcyj-msxm2388), the Key Technology Innovation and Application Demonstration Projects in Chongqing (Grant no. cstc2018jszx-cyzdX0068), and the Major Research and Development Projects on Artificial Intelligence Technology Innovation in Chongqing (Grant no. cstc2017rgzn-zdyfX0003).

References

[1] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys*, vol. 42, no. 3, pp. 1–42, 2010.

[2] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic QoS management and optimization in service-based systems," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2011.

[3] M. Malek, "Predictive analytics: a shortcut to dependable computing," in *Proceedings of the 9th International Workshop on Software Engineering for Resilient Systems*, pp. 13–17, Serene, Geneva, Switzerland, August 2017.

[4] H. Wang, L. Wang, Q. Yu, Z. Zheng, and Z. Yang, "A proactive approach based on online reliability prediction for adaptation of service-oriented systems," *Journal of Parallel and Distributed Computing*, vol. 114, pp. 70–84, 2018.

[5] Z. Ding, T. Xu, T. Ye, and Y. Zhou, "Online prediction and improvement of reliability for service oriented systems," *IEEE Transactions on Reliability*, vol. 65, no. 3, pp. 1133–1148, 2016.

[6] C.-H. Lung, X. Zhang, and P. Rajeswaran, "Improving software performance and reliability in a distributed and concurrent environment with an architecture-based self-adaptive framework," *Journal of Systems and Software*, vol. 121, pp. 311–328, 2016.

[7] H. Okamura, J. J. Zheng, and T. Dohi, "A statistical framework on software aging modeling with continuous-time hidden Markov model," in *Proceedings of the 2017 IEEE 36th International Symposium on Reliable Distributed Systems, Symposium on Reliable Distributed Systems*, pp. 114–123, IEEE, New York, NY, USA, September 2017.

[8] M. Silic, G. Delac, and S. Srbljic, "Prediction of atomic web services reliability based on *K*-means clustering," in *Proceedings of the 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 70–80, ESEC/FSE 2013, Saint Petersburg, Russia, August 2013.

[9] J. Zhu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "CARP: context-aware reliability prediction of black-box web services," in *Proceedings-2017 IEEE 24th International Conference on Web Services*, pp. 17–24, ICWS, Honolulu, HI, USA, June 2017.

[10] S. Malefaki, S. Trevezas, and N. Limnios, "An EM and a stochastic version of the EM algorithm for nonparametric hidden semi-Markov models," *Communications in Statistics—Simulation and Computation*, vol. 39, no. 2, pp. 240–261, 2010.

[11] Q. Guan, Z. Zhang, and S. Fu, "Ensemble of Bayesian predictors and decision trees for proactive failure management in cloud computing systems," *Journal of Communications*, vol. 7, no. 1, pp. 52–61, 2012.

[12] Z. Xue, X. Dong, S. Ma, and W. Dong, "A survey on failure prediction of large-scale server clusters," in *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pp. 733–738, SNPD, Qingdao, China, August 2007.

[13] J. Diebolt, M. Garrido, and C. Trottier, "Improving extremal fit: a Bayesian regularization procedure," *Reliability Engineering & System Safety*, vol. 82, no. 1, pp. 21–31, 2003.

[14] S. Miao, K. Xie, H. Yang, R. Karki, H.-M. Tai, and T. Chen, "A mixture kernel density model for wind speed probability distribution estimation," *Energy Conversion and Management*, vol. 126, pp. 1066–1083, 2016.

[15] A. Amin, L. Grunske, and A. Colman, "An automated approach to forecasting QoS attributes based on linear and nonlinear time series modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 130–139, IEEE ACM International Conference on Automated Software Engineering, Essen, Germany, September 2012.

[16] A. Amin, L. Grunske, and A. Colman, "An approach to software reliability prediction based on time series modeling," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1923–1932, 2013.

[17] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models," in *Proceedings of the2012 IEEE 19th International Conference on Web Services*, pp. 74–81, ICWS, Honolulu, HI, USA, June 2012.

[18] H. Wang, L. Wang, Q. Yu, Z. Zheng, A. Bouguettaya, and M. R. Lyu, "Online reliability prediction via motifs-based dynamic Bayesian networks for service-oriented systems," *IEEE Transactions on Software Engineering*, vol. 43, no. 6, pp. 556–579, 2017.

[19] H. Wang, Z. Yang, Q. Yu, T. Hong, and X. Lin, "Online reliability time series prediction via convolutional neural network and long short term memory for service-oriented systems," *Knowledge-Based Systems*, vol. 159, pp. 132–147, 2018.

[20] Z. Zheng and M. R. Lyu, "Personalized reliability prediction of web services," *ACM Transactions on Software Engineering and Methodology*, vol. 22, no. 2, 2013.

[21] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2013.

[22] M. Kubacki and J. Sosnowski, "Multidimensional log analysis," in *Proceedings of the 12th European Dependable Computing Conference*, pp. 193–196, EDCC, Gothenburg, Sweden, September 2016.

[23] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "A lightweight algorithm for message type extraction in system application logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 11, pp. 1921–1936, 2012.

[24] M. Kubacki and J. Sosnowski, "Applying time series analysis to performance logs," in *Proceedings of the SPIE 9662, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments*, pp. 1–12, Wilga, Poland, May 2015.

[25] I. Iguyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[26] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[27] S. Chatterjee and A. Roy, "Novel algorithms for web software fault prediction," *Quality and Reliability Engineering International*, vol. 31, no. 8, pp. 1517–1535, 2015.

[28] A. Graves, *Long Short-Term Memory*, Springer, Berlin, Germany, 2012.

[29] H. Goel, I. Melnyk, N. mOza, B. Matthews, and A. Benerjee, *Multivariate Aviation Time Series Modeling: VARs vs. LSTMs*, https://goelhardik.github.io/images/Multivariate_Aviation_Time_Series_Modeling_VARs_vs_LSTMs.pdf, 2016.

[30] S. Seto, W. Zhang, and Y. Zhou, "Multivariate time series classification using dynamic time warping template selection for human activity recognition," in *Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence*, pp. 1399–1406, SSCI, Cape Town, South Africa, December 2015.

[31] G. A. Hoffmann, K. S. Trivedi, and M. Malek, "A best practice guide to resource forecasting for computing systems," *IEEE Transactions on Reliability*, vol. 56, no. 4, pp. 615–628, 2007.

[32] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world web services," in *Proceedings of the 8th International Conference on Web Services*, pp. 83–90, ICWS, Miami, FL, USA, July 2010.

[33] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2014.