Quality of Service Forecasting with LSTM Neural Network

Tomas Jirsik*†, Štěpán Trčka*, Pavel Celeda*
*Institute of Computer Science, †Faculty of Informatics
Masaryk University, Brno, Czech Republic
jirsik@ics.muni.cz, 445586@mail.muni.cz, celeda@ics.muni.cz

***Abstract*—A robust and accurate forecast of the Quality of Service (QoS) attributes is essential for effective web service recommendation, enhanced user experience, and service management. Deep learning methods, especially Long Short-Term Memory Neural Networks (LSTM NN), have proven to be worthy for sequence forecasting in various domains recently. In this paper, we pilot an experimental application of LSTM NN in the domain of QoS forecasting. We develop a LSTM NN model for QoS prediction and compare its forecast performance with existing approaches for QoS attribute forecasting – ARIMA and Holt-Winters models. The approaches are compared on two real-world QoS attribute datasets created using centralized passive QoS attribute collection technique. Our results show that LSTM NN improves the accuracy of QoS forecast for attributes collected with high granularity while maintaining a reasonable computation time.**

I. Introduction

Web services (WSs) are an internal part of modern IT systems and applications nowadays because of their re-usability, composability, and global accessibility via the Web. The large number of the WSs has resulted in a shift in the web search and recommendation practice. Given the high number of WSs, it is highly probable, that there exist two functionality-equivalent WSs. Therefore, it has become a convention to search and recommend the WSs based on non-functional properties [1]. Among other non-functional properties studied, the *Quality of Service* (QoS) is the most widely considered [2].

Quality of Service is an abstract term that is derived from measurable QoS attributes [3]. The QoS attributes, such as application response time (RT), or network transport time (NTT), are highly volatile in time. [4]. The high volatility of the QoS attributes represents a challenge for web service search and recommendation systems. The frequency of the QoS attributes updates is usually not high enough to provide up-to-date data for reliable service recommendation. Moreover, the measurement of QoS attributes by service providers has shown to be unreliable or service degrading [5]. To obtain an accurate, up-to-date QoS attributes for WSs recommendation, most existing studies leverage time series forecasting [2].

A challenge for QoS attributes forecast is to determine, which time series forecasting method is the most suitable for forecasting the WSs QoS attributes. The suitability is determined from the performance of the forecasting methods, mostly forecast accuracy. Syu et al. [2] provided a comprehensive empirical study on the comparison of available methods for QoS attribute forecasting. Since the publication of the study, new promising methods for time series forecasting have been introduced, though. Most notably, *deep learning* methods, especially *Long Short-Term Memory Neural Networks* (LSTM NN), show promising results in time series forecasting tasks in other domains [6]. To the best of our knowledge, the application of these novel approaches to QoS attributes forecast tasks, and their evaluation are still missing.

In this paper, we aim to address this niche and evaluate the suitability of LSTM NN for QoS forecasting. We compare the forecast accuracy of LSTM NN with the accuracy of two other methods commonly used for time series forecasting – ARIMA and Holt-Winters model. Apart from the forecast accuracy comparison, we investigate the time cost of the model creation. The method is evaluated on five different QoS attributes measured for two WSs at two different levels of granularity. To create the QoS attribute's time series, we employ a centralized passive collection of QoS attributes that enables transparent continuous collection of QoS attributes even in large-scale high-speed networks and does not create a negative impact on the monitored services. The QoS attribute time series, source code, and experiment results are publicly released at [7] to make our research reproducible.

The main contributions of our work are threefold:

- introduction of the method for centralized collection of QoS attributes,
- development of LSTM NN model for QoS attribute forecast,
- evaluation of the LSTM NN model performance (precision and time complexity) at real-world QoS attributes time series.

This paper is organized as follows. In Section II, we present a centralized collection of QoS attributes based on the passive observation of network traffic. Next, we describe a theoretical background of the selected methods for time series forecasting in Section III. Experiment methodology, dataset description, and forecast models description are outlined in Section IV. Next section reveals the actual settings of the forecasting models estimated the provided dataset. Section VI presents the results of the forecast evaluation of the selected methods. The results are discussed in Section VII. Section VIII overviews the relevant related works, and Section IX concludes the paper.

978-3-903176-15-7

II. Centralized QoS Attribute Collection

The QoS attributes are highly volatile in time [4]. Hence, WSs search and recommendation services require most recent, near real-time measurements of QoS attributes. A straightforward solution to achieve an up-to-date data is an active measurement of QoS attributes. A system for active QoS measurement sends out periodical service invocations and determines the QoS attribute values based on a service reply. The service invocation approach to QoS attribute measurement provides an accurate data on the one hand. On the other hand, the invocation suffers from several disadvantages. Since frequent observations are required, the invocation of services produces extra overhead for the service. Further, monitoring of several services (e.g., web instances in a cloud) further increases the number of invocations necessary, which adds further costs to QoS measurement. Apart from this approach, there is still a lack of approaches on how to obtain a frequently updated QoS information [5].

We show that an alternative approach to active QoS attribute collection is *IP flow network monitoring*. Majority of the QoS attributes can be obtained from packet-level network analysis. It is possible to determine the QoS attributes from packet key values, e.g., timestamps, TCP flags, payload sizes, fragmentation, and so forth. Architectures for flow traffic measurement mainly serve to provide network visibility, for network reporting, and security measurements. These architectures are designed to measure traffic flows even in high-speed, large-scale networks [8]. A flow measurement architecture consists of network *probes* and *collectors*. A probe observes packets at an observation point in a network and aggregates the packets into *IP flows*. All packets in an IP flow have the same flow keys, e.g., source and destination IP, port and protocol [9]. The IP flows are collected, stored, and analyzed at a collector.

Modification of network probes used for network traffic monitoring, so that they can observe packet keys needed for QoS attribute computation, allows for passive measurement of QoS attributes even at large-scale and at high speeds. Such a modification of probes enables us to monitor, among others, application response time, network transport time, transaction size, packet fragmentation, and jitter. These attributes can be measured for all client-service communications observed in a network. This approach enables centralized monitoring of WSs performance with high granularity at a large scale. Moreover, the network traffic flow measurement is transparent and do not add any overhead at either the client or server side as the data are monitored in a network. Beyond that, we can measure a number of concurrent users of the service, which would be impossible in case of active service invocation used for QoS measurement.

The optional location of flow observation points anywhere in a network allows for precise QoS measurement. If an accurate measurement of the user experience is required, we place an observation point close to the client-side, and the observations include both server and network response times. In case only server response time is of interest, the observation point is placed close to the server. The location of an observation point at the network perimeter allows for QoS monitoring with the purpose of maintaining the service level agreements.

III. Forecasting Methods

Time series analysis and forecasting can be performed using a wide range of algorithms. For the forecast of QoS attributes, we choose to employ AutoRegressive Integrated Moving Average (ARIMA) model, Holt-Winters Exponential Smoothing (Holt-Winters), and Long Short-Term Memory Neural Networks (LSTM NN). The ARIMA and Holt-Winters are methods frequently used for forecasting in QoS and network traffic analysis domain [4, 10]. Therefore, these methods can be naturally used as a baseline for comparison of the LSTM NN forecast performance. In this section, we describe these selected methods.

A. ARIMA Model

ARIMA model is based on AutoRegressive Moving Average (ARMA) model proposed by Box and Jenkins [11]. The main assumption for the ARMA model is the stationarity of the time series. The ARIMA model overcomes this limitation by converting a non-stationary time series into a stationary one via *differencing* of time series – a stationary time series is created by computing the differences between consecutive observations of a time series.

The $ARIMA(p, d, q)$ model is defined as

$$(1-\phi_1 B-\cdots-\phi_p B^p)(1-B)^d y_t = c+(1+\theta_1 B+\cdots+\theta_q B^q)\varepsilon_t \quad (1)$$

where ε_t is white noise and B is *backshift* operator defined as $By_t = y_{t-1}$. The autoregressive polynomial of order p is $AR(p) = (1 - \phi_1 B - \cdots - \phi_p B^p)$ and the moving average polynomial with order q is $MA(q) = (1 + \theta_1 B + \cdots + \theta_q B^q)$. The parameter d is a differencing order and formula $(1 - B)^d$ represents the differencing of the time series. The autoregressive coefficients $\phi = (\phi_1, \cdots, \phi_p)^T$ and moving average coefficients $\theta = (\theta_1, \cdots, \theta_q)^T$ can be estimated using statistical approaches, e.g., the least square method or the maximum likelihood.

B. Holt-Winters

Holt-Winters forecasting procedure belongs to the univariate projection methods where forecasts are based only on the past values of the forecasted variable. The Holt-Winters approach generalizes simple exponential smoothing to deal with a trend and seasonal variation of a time series [12]. Compared to the previous model, it can be initialized with a fewer data.

There exist two types of Holt-Winters model - *additive* and *multiplicative*. The additive model assumes a constant size of the seasonal component of a time series, while the multiplicative model assumes a proportional seasonal component to the deseasonalized mean level. The additive seasonal Holt-Winters model is defined as [13]:

$$L_t = \alpha(y_t - I_{t-p}) + (1-\alpha)(L_{t-1} + T_{t-1}) \quad (2)$$

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1} \quad (3)$$

$$I_t = \gamma(y_t - L_{t-1} - T_{t-1}) + (1-\gamma)I_{t-p} \quad (4)$$

where $0 < \alpha, \beta, \gamma < 1$ are smoothing parameters for mean level L_t, trend T_t and seasonal index I_t, and p is a season length. For a definition of the multiplicative model see [13].

A new forecast for y_t k periods ahead is given by

$$\hat{y}_t(k) = L_t + kT_t + I_{t-p+1+(k-1)modL} \quad (5)$$

Initial values for level, trend, and seasonal component can be chosen randomly or by the average of the early observations. The model parameters α, β, γ are can be estimated using the least square method, the maximum likelihood method, or can be based on the Akaike's (AIC) or Bayesian (BIC) Information Criteria.

C. LSTM Neural Network

Long Short-Term Memory Neural Network (LSTM NN) belongs to the family of recurrent neural networks (RNN). RNN can send their output back to previous layers which allows information to persist. Although RNNs can handle long-term dependencies in theory, the practical evaluation shows that RNN does not seem to be able to capture them [14]. LSTM NN was proposed by Hochreiter et al. in [6] to address this issue of RNN. LSTM NN introduces novel single cell layers that can successfully solve long time lag tasks that have not been solved by RNN before.

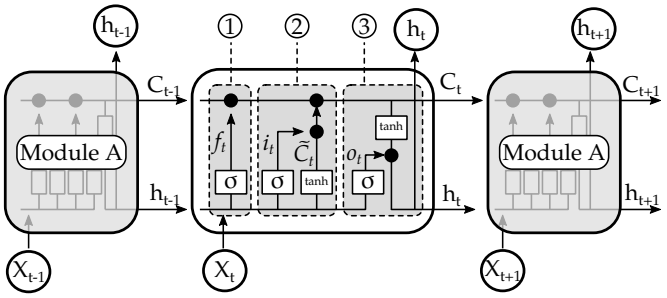


Fig. 1: LSTM Neural Network [15]

The schema of the LSTM NN is depicted in the Figure 1. A RNN is a chain of repeating modules with a single layer. The LSTM NN maintains the chained structure and introduces interconnected multiple layers in a single module. Each repeating module is represented by a cell state C_t. Each layer is able to add or remove information from a cell state via regulatory gates (①, ②, ③).

The first gate ① is called the *forget gate*. This layer contains a function that decides which elements of the state C_{t-1} are kept or forgotten based on input X_t and h_{t-1}. Second gate ②, the *input gate*, decides which values are updated i_t, and creates the candidate vector of values $\tilde{C}_t$. The new cell state is then computed as

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \quad (6)$$

The *output gate* ③ defines outputs of the cell. First, a function o_t decides which parts of the cell state C_t are used for output. Next, the cell state C_t is transformed via $\tanh$ function and the final output $h_t = o_t * \tanh C_t$ is computed. The cell state C_t and output value h_t serve as an input for a next module iteration. The LSTM NNs is iteratively trained on training data using backpropagation algorithms. The weights and parameters are optimized using optimization algorithms and loss functions.

The LSTM NNs have recently gained attention as world-leading vendors used them in their product, such as Google for Google Translator [16], Apple for Quicktype keyboard [17]. We believe that the properties of LSTM NN, especially the ability to solve long time lag tasks, fits tasks for QoS attribute forecasting and can increase the precision of QoS forecasts.

IV. Experiment Methodology

This sections provides a detailed description of the experiment setup including the dataset description, construction of the forecasting models, and performance evaluation methodology.

A. Dataset

We collected QoS attributes for two public web services. The first monitored service (SERV-1) is a portal for electronic information resources available at our university. This portal serves as a search engine for licensed resources for science, research, and teaching provided for our students. The second monitored service (SERV-2) is a web presentation of the Faculty of Science. While the first service is dynamic and interactive, the second one represents a static web presentation.

Both services were monitored over a month period from June 16th to July 15th, 2018. Using the approach described in Section II, we collected QoS attributes listed in Table I. *Number of concurrent users* denotes the number of users visiting a page in parallel in an observation window. *Application response time* denotes the time between a query and its response. This attribute includes both network response time and time needed for query processing on the server side. *Transaction count* represents a number of unique transactions denoted by URL path. *Network transport time* is defined as the time for which a transaction is transferred over a network, and *Transport size* shows the size of transactions observed in a given window. For each attribute, we measured various statistics such as count, minimum, maximum, average, values of 50-th, 90-th, and 99-th percentile (p50, p90, and p99).

TABLE I: Monitored QoS Attributes

QoS Attribute	Measured Statistics
Number of concurrent users (USR)	**count**
Application response time (s) (ART)	min, max, **avg**, p50, p90, **p99**
Transaction count (TC)	**count**
Network transport time (s) (NTT)	min, max, **avg**, sum
Transport size (s) (TS)	min, max, avg, **sum**

The attribute's statistics that enter the experiment are highlighted in Table I in bold. For ART attribute, we choose to use the average and value of the 99-th percentile for all transaction in a given time window. The average is a typical statistics computed in the QoS domain. The 99-th percentile is chosen to test the forecast of outlying values of ART, which could be used for the forecast of ART depreciation. For NTT attribute, we select average over all transactions. TS is more suitably represented by a sum of transport size of all transaction than by average transaction size, which does not reflect the number of transactions. The samples of the measured QoS attribute time series are provided in Figure 2.

The time series were collected in two different granularity settings - 5 minutes and 1 hour. The 5-minute granularity represents a normal update interval used in network monitoring domain. The 1-hour granularity still provides a sufficient number of observations per day, on the one hand, and adequate level of aggregation to both smooth a time series and maintain main series characteristics, on the other hand. Missing observations in time series (1.15% of all observations) were substituted by the linear interpolation.

The dataset was split into training and testing dataset. The training dataset contains one week period and testing dataset covers the remaining three weeks. The training dataset contains enough data to capture day-night seasonal pattern commonly observed at several QoS attributes, e.g., number of users. The trained models are expected to capture this pattern. Once a model is trained on the training dataset, a testing dataset is used for model performance evaluation. The size of the testing dataset enables us to evaluate even a short-term forecast performance of the models.

B. Forecast

We describe the taxonomy of forecast types in this section and identify the forecast type used for QoS attribute forecasting in this paper. The taxonomy categorizes the forecast by the time scale, number of forecasted observation and by the forecast frequency.

The forecast can be classified into following four categories, depending on the time scale [18]: *real-time*, where forecast does not exceed several minutes, *short-term*, where values for one to several hours are forecast, *middle-term*, for several days long forecast, and *long-term*, where forecasts for more than several months or years belong.

Another approach to forecast classification is to differentiate the forecasts according to the number of forecasted observation, regardless of the predicted timescale. *One-step forecast* predicts one observation ahead, while *multi-step* predicts more than one observation points from the training data.

Last, the forecast can be classified by the frequency of the forecasts. *One-time* forecast is produced only once and usually serves for static, batch-based tasks. *Continuous* forecast produces new forecasts for a given scale each time a new observation is available.

In our paper, we focus on *continuous*, *one-step* forecast that covers *real-time* scale, when considering five-minute granularity, and *short-time* scale in case of one-hour granularity of a time series.

C. ARIMA Model Construction

We construct $ARIMA(p, d, q)$ model according to the Box-Jenkins methodology [19]. The model is constructed in the following steps.

First, we estimate the differencing parameter d. We incrementally increase the differencing order of a time series. Each differenced time series is tested for stationarity using the Augmented Dickey-Fuller test. Once a differenced time series passes the test, the differencing parameter d is set as the lowest differencing order of the tested time series.

Second, we estimate the autoregressive and moving average parameters p, q. We plot an autocorrelation plot (ACF) and a partial autocorrelation plot (PACF) for each time series to identify parameters p and q, respectively. The ACF is plotted up to 250 lag order. The window for PACF contains 50 observations. The parameters are identified from spikes in the plots. In case more parameter combinations are identified, we employ AIC to select the best combination of the model parameters.

Third, we estimate a model representing the time series. The model is fitted by exact maximum likelihood using Kalman Filter. First, we maximize the conditional sum of squares likelihood. Next, we use the conditional values as starting values for the computation of the exact likelihood via the Kalman filter. The maximum number of iterations of the function evaluation is set to 500.

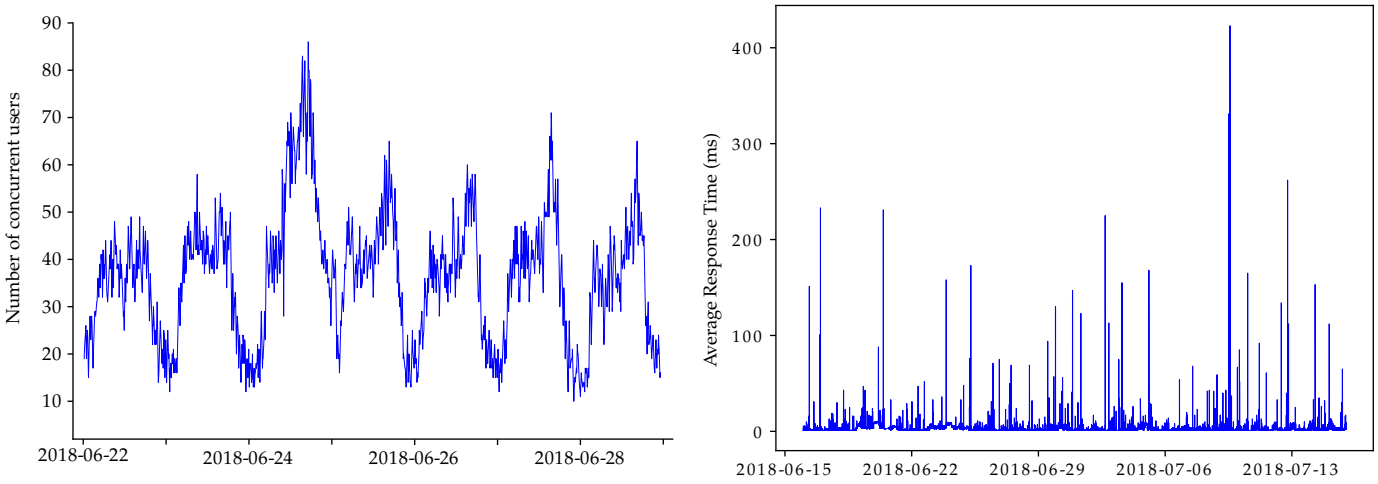
The model fit is performed for each new forecast. Once a model is fitted, a one-step, continuous forecast is computed from the model.

D. Holt-Witers Model Construction

The first step of Holt-Winters model construction is the identification of the model type – additive or multiplicative. The identification of the model type is based on manual inspection of the seasonal and trend components of a time series. If the seasonal component increases in the exponential fashion, the model is multiplicative, otherwise the additive model is applied. For the identification of the level, trend and seasonality presented in a time series, we employ seasonal decomposition using moving averages.

Second step is to identify a season length using both information from ACF, PACF, and seasonal decomposition of a time series. When estimating the season length, we take into account the real-world properties of the network traffic, such as day-night and week patterns.

Third, the model is fitted by maximum likelihood estimation. As we expect the time series to represent the additive model, the maximum likelihood estimation provides the same results as the minimization of the sum of squares. We do not pose any restriction on the model parameters (except for the case of missing trend or seasonal component), nor we set any initial values of the model's parameters. The final model is chosen based on the corrected AIC criterion.



(a) Number of concurrent users (one-week sample)

(b) Average response time

Fig. 2: QoS attributes time series (SRV-1, 5-minute granularity)

E. LSTM Neural Network Model Construction

The LSTM NN networks are sensitive to large scaled values in data. According to [20], the data should be rescaled when a sigmoid activation function is used. Hence, we rescale the time series to the interval $< 0, 1 >$.

We implement a RNN that consists of three layers – input, hidden, and output. For one step prediction, we need to have exactly one input and one output node. The number of hidden layers is determined by an experimental evaluation concerning the time complexity of the model training.

The hidden layer consists of LSTM cells with the sigmoid activation function, which reflects initial data transformation. The NN is configured to use Mean Square Error (MSE) as the stop loss function that is minimized in the backpropagation process during the model fit. The backpropagation process uses the Stochastic Gradient Descent (SGD) optimizer with default parameters to achieve the optimal accuracy of the model. The number of iteration for LSTM NN is experimentally derived from the learning curve of the NN. The model is fitted on the training dataset and is further used for prediction on the training dataset.

F. Performance Evaluation

We evaluate the performance of the LSTM NN with respect to forecast accuracy and time needed to fit a model. To compare the forecast accuracy of the LSTM NN with other approaches, we employ *Mean Absolute Percentage Error* (MAPE) metrics. We prefer MAPE to the commonly used *Mean Absolute Error* (MAE) or Mean Squared Error (MSE), as MAPE abstracts from the scale of time series and allows for comparison of QoS attributes prediction regardless the popularity of the service. The MAPE is computed as follows:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| * 100 \qquad (7)$$

where y_t is the observed value and $\hat{y}_t$ is the forecasted value provided by a given model.

The time needed to fit a model is measured from the start of the model fitting process and ends when the best fit for a model is found. The data preparation tasks, such as missing values interpolation, are not included in this measurement, as well as the time needed to compute the forecast. We choose to compare only the time needed to fit a model, as it is the most computationally demanding task. Once a model is estimated, the actual forecast can be computed from the fitted model nearly instantly. The time cost was measured on a machine with 6 AMD Ryzen5 CPUs ∼3.8GHz, 6GB RAM, and 50GB HDD.

V. Forecast Models Settings

The methodology for model construction described in the previous section was applied to the training dataset to estimate the models for the forecast. This section describes the properties of the time series of measured QoS attributes that are relevant for the model estimation. Next, we present the models estimated on the training data set that we use for the forecast evaluation.

A. Properties of QoS Attribute Time Series

The measured time series can be divided into two separate clusters – a time series showing a diurnal seasonality and the time series without the seasonality. The USR and TC time series show a strong diurnal pattern (see Figure 2a). We can differentiate working hours showing an increase in user and transaction counts and night hours with a significant drop in counts. Besides day-night pattern, we observe working days–weekend seasonality. The number of transactions is lower during a weekend compared to business days. Moreover, we observe a strong positive correlation between these time series.

The ART, NTT, and TS time series do not show any seasonality as the latter time series (see Figure 2b). It shows that

the monitored services and associated network infrastructure have enough resources at disposal so that an increase in the number of the transaction does not slow down network or service and does not increase response time. We expect that ART, NTT, and TS time series will be difficult to forecast with high precision, as they do not contain any observable pattern.

B. ARIMA

We estimated the $ARIMA(p, d, q)$ according to the methodology described in Section IV. The estimated parameters are presented in Table II.

TABLE II: ARIMA Models Settings

QoS Attribute	SERV-1		SERV-2	
	5 min	1 h	5 min	1 h
Number of concurrent users (USR)	(2,0,0)	(2,0,0)	(2,0,0)	(2,0,0)
Response time - avg (ART-avg)	(2,1,0)	(1,0,0)	(1,0,0)	(1,0,0)
Response time - p99 (ART-p99)	(1,1,0)	(2,1,0)	(0,0,1)	(1,0,0)
Transaction count (TC)	(3,0,0)	(2,0,0)	(4,0,0)	(3,0,0)
Network transport time (NTT)	(2,1,0)	(1,1,0)	(0,0,1)	(1,0,0)
Transport size (TS)	(3,0,0)	(2,0,0)	(3,0,0)	(3,0,0)

We observe that USR, TC, and TS time series pass the stationarity test and no differencing is necessary for both monitored services. One order differencing was necessary for other time series to fulfill the stationarity. Further, our estimation shows, that the autoregressive part (AR) of the ARIMA model is sufficient in the majority time series to describe a time series. This fact is given by an ability of AR to capture the anomalies presented in time series. In several cases, the applied Box-Jenkins methodology suggested the MA model. The automated testing of other ARIMA parameters combinations showed better AIC results for AR model, though, due to anomalies presented in the time series.

C. Holt-Winters

The Holt-Winters model required to identify seasonality and the season length. The decomposition of the time series showed a clear additive pattern in seasonality present in the time series. Hence, we trained the additive Holt-Winters model. As described above, USR and TC time series contains both day-night and working day-weekend seasonality.

We set the season length for one week, as this period covers both seasonality. If we set seasonality to one day, the weekend seasonality would not be reflected, and the prediction performance would decrease.

The maximum likelihood estimation of the model's parameters confirmed the seasonality presented in USR and TC time series. The γ parameters were non-zero, and the model accented the recent observations at time series. The $beta$ parameters were zero for all time series as there is no trend present in the time series. The estimation α parameter varied from 0.05 to 0.99 over all time series.

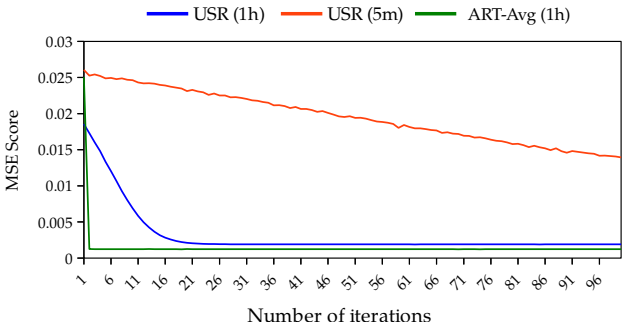


Fig. 3: Learning Rates of LSTM NNs

D. LSTM Neural Networks

For the LSTM NN, we need to set a number of LSTM cells in the hidden layer and the maximum number of iteration for backpropagation process. Based on our experience, we include two LSTM cells into the NN. Our experience shows that a higher number of the cells increases the forecast performance only incrementally while the time needed to fit the model increases significantly.

We derived the maximum number of iterations using learning rate curves of the NN. The learning rates of the NNs for the selected time series are presented in Figure 3. In general, the learning rates of ART, NTT, and TS time series drop rapidly to zero after two iterations. The USR and TC time series with one-hour granularity show only minor improvement in learning after 21 iterations. The learning curve of USR and TC time series with five-minute granularity has a linear decreasing trend.

Based on the above analysis, we set the maximum number of iterations to 100, which allows an optimal fit of NN for the majority of the time series, on the one hand, and does not increase the time needed for the model fit beyond the reasonable amount, on the other hand. We set the same values for the number of cells and the number iterations for all models to be able to compare the model forecast performances.

VI. Experiment Results

We use the fitted model for the one-step continuous forecast on the testing dataset and evaluate the forecast accuracy using MAPE as described above. The results of our evaluation are presented in Table III. The best forecast accuracy for five-minute granularity is highlighted in bold and for one-hour granularity in italics.

The LSTM NN outperforms other methods mainly in forecast accuracy of time series with high granularity. Given the minute granularity, the LSTM NN show the best forecasting accuracy in $58.3\,\%$ of time series. Given the one hour granularity, the LSTM NN surpass other methods in $50\,\%$ of time series. The Holt-Winters shows the weakest performance. The ARIMA model is comparable with LSTM NN in the prediction of time series with one-hour granularity.

LSTM NN shows the best performance for five-minute USR and ART-avg time series. The forecast of these time

TABLE III: Mean Absolute Percentage Error for QoS Attributes Forecast

QoS Attribute	Service	ARIMA		Holt-Winters		LSTM NN	
		5 min	1 h	5 min	1 h	5 min	1 h
Number of concurrent users (USR)	SERV-1	7.79	*13.70*	28.09	38.89	**2.16**	20.27
	SERV-2	5.44	*10.02*	24.41	32.11	**1.61**	20.84
Response time - avg (ART-avg)	SERV-1	119.04	*113.01*	212.61	141.45	**100.99**	116.52
	SERV-2	103.44	41.24	66.48	45.08	**40.39**	*30.87*
Response time - p99 (ART-p99)	SERV-1	**250.42**	*110.83*	504.83	195.52	497.23	153.43
	SERV-2	205.54	84.62	165.77	126.70	**106.58**	*71.70*
Transaction count (TC)	SERV-1	**76.28**	*50.80*	310.98	272.62	252.68	119.89
	SERV-2	36.23	28.75	226.91	198.95	**28.07**	*11.40*
Network transport time (NTT)	SERV-1	288.63	96.53	**238.26**	99.16	460.22	*69.96*
	SERV-2	**374.92**	81.40	394.67	96.04	409.73	*34.31*
Transport size (TS)	SERV-1	46.82	*25.99*	51.79	160.12	**46.40**	39.72
	SERV-2	**112.90**	48.84	210.06	154.127	386.56	*35.73*

series with one-hour granularity is dominated by ARIMA forecasting model. We recall that USR time series shows a diurnal pattern and the RT-avg is a smoothed time series. The one-hour granularity of these time series generates a smooth enough time series so that ARIMA can capture the time series behavior. However, when the granularity of these time series increases to five minutes, the time series become more volatile. In this case, LSTM NN can better learn the time series model despite the increased volatility.

The values of MAPE varies from 1.61 to 504.83. The lowest MAPE is achieved from five-minute USR and TC time series. The high values of MAPE in ART-p99 time series is caused by the presence of several outliers in the testing dataset, which increases the error mean.

The measurement of time needed to fit a model highlights the complexity of LSTM NN. The overall results are presented in Table IV. The fitting time measured for five minute granularity time series is naturally higher than the one-hour time series as more observation needs to be fitted. The increase of time needed to fit Holt-Winters and LSTM NN models at five-minute time series compared to one-hour time series is proportional to the increase in the number of observation needed to fit. The ARIMA model shows an overhead when fitting more observation points.

The Holt-Winters method shows the fastest model construction as the model is constructed using only maximum likelihood estimation compared to ARIMA where additional optimization techniques are present. Surprisingly, LSTM NN model is fitted faster than ARIMA model in five-minute granularity. We explain this result by the employment of two-step optimization used during ARIMA fit.

TABLE IV: Model Estimation Duration (s)

Granularity	ARIMA	Holt-Winters	LSTM NN
5 minutes	574.56 ± 509.06	44.04 ± 4.17	397.48 ± 43.63
1 hour	30.21 ± 30.42	2.92 ± 0.82	33.70 ± 1.61

VII. Discussion

In this section, we discuss several aspects of the application of LSTM NN in our comparison. We inspect the selection of the initial weights for LSTM NN closely, look into the evaluation metric, and investigate possibilities of the improvement of LSTM NN speed and accuracy.

The value of MAPE for LSTM NN is influenced by the configuration of the initial weights. According to the described methodology, we set the initial weights at random. Hence, we investigated the influence of the random initial weights of LSTM NN on the values of MAPE to capture a possible bias introduced to our experiment. We ran LSTM NN forecast twenty-times with random initial weights and computed the MAPE distribution. Sample MAPE distribution is presented in Figure 4. On average, the variability of the MAPE is $12\,\%$ of its mean value. Nevertheless, even if we include this bias in the comparison of LSTM NN with other methods, the method's ranking holds.

As stated above, the high values of the MAPE for NTT time series are caused by the outliers in the observations. To omit the influence of the outliers to forecast accuracy, we try to use an alternative metrics for forecast accuracy evaluation called Symmetric Mean Absolute Percentage Error $SMAPE = \frac{100}{n}\sum_{t=1}^{n}\frac{|\hat{y}_t - y_t|}{(|\hat{y}_t|+|y_t|)/2}$ that is better protected against outliers than MAPE. Using SMAPE, the error of NTT decreased from 460.22 to 90.98 in LSTM NN forecast.

Last, we investigated the possibilities of improvement of the fitting procedure of the LSTM NN. A substantial portion of time during a LSTM NN fit is consumed by LSTM optimizer searching for the minimum of the loss function. Similarly, the forecast accuracy is dependent on the optimizer performance. SGD optimizer that we used in our experiment does show slow progress towards a minimum [21]. The authors of [21] suggest that implementation of Adam or RMSProp optimizer would

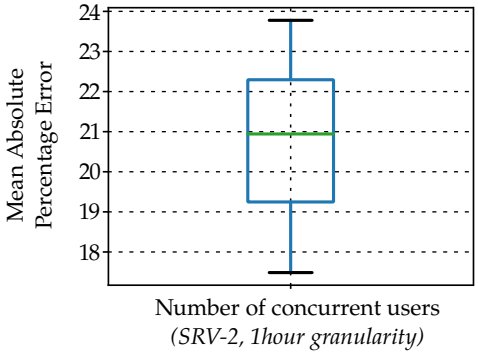


Fig. 4: LSTM NN – MAPE variability

increase the speed of minimization and the overall forecast performance of LSTM NN.

The MAPE values demonstrate the suitability of the various QoS attributes for forecasting. The MAPE values for NTT and ART-p99 are too high to be used for WSs recommendation based on QoS prediction. Number of the concurrent users is an attribute that can be predicted with high probability. Average response time varies by the type of service. Note, that the data were not preprocessed in any way. Having the data preprocessed (e.g. log transformation), we could possible have achieved lower error rates.

VIII. Related Work

Suitability of various time series analysis methods for QoS prediction has already been surveyed in literature. Cavallo et al. [22] provided an empirical comparison of several methods for one-step-ahead prediction of QoS attribute, including a naive approach based on current value, average, linear prediction, and ARIMA. The compared methods were evaluated on a dataset that captured QoS attributes for ten services. This dataset was later used by Syu et al. in even more extensive comparisons of time series analysis methods for QoS forecasting [2, 23]. The authors compared all major approaches to time series forecasting, including regression, ARIMA, exponential smoothing, GARCH models, neural networks and genetic programming methods. The authors concluded that there is no single most suitable method for dynamic QoS forecasting. For the highest accuracy of the forecast, they recommend the genetic programming method, for quick prediction, the naive method is suggested. The LSTM NN were not mentioned in any of these surveys, though.

Apart from surveys, several improvements of existing time series forecasting methods have been suggested to improve QoS forecast accuracy. Wang et al. [24] proposed to use a spatial-temporal QoS prediction used for WSs recommendation. Geolocation of a WSs was employed to reduce the number of candidate WSs and to improve the forecasting accuracy. Amin et al. [4] used a forecasting approach based on the combination of ARIMA and GARCH model that was able to capture the QoS attribute's volatility and improve the accuracy of the forecast. The authors also investigated the automated approach to QoS forecasting in [25]. A linear and non-linear modeling methods were used to create models for QoS prediction automatically, without human intervention. For this use-case, their approach improved the forecasting accuracy on by average 35 %. Zhu et al. [5] employed an adaptive matrix factorization to perform online QoS prediction for candidate services for runtime service adaptation. They evaluated their approach regarding accuracy, efficiency, and robustness. Further, Li et al. successfully used Holt-Winters model to forecast web pages views in [10].

We identified a few publications that already used neural networks for QoS predictions. Senivongse et al. [26] used artificial neural networks for QoS prediction that assist the service composition with optimal overall QoS. Zadeh and Seyyedi [27] leveraged feed-forward NN with sigmoid activation function for QoS forecasting. They reported a wide MSE range from 572 to 2393372. The approach was not compared with any other time series prediction methods, though. Bendriss et al. [28] used RNN for service level objectives breaches.

IX. Conclusions

This paper displays an experimental application of Long Short-Term Memory Neural Networks in the QoS forecast domain. We present a performance comparison of LSTM NN with two commonly used time series forecasting methods ARIMA and Holt-Winters on a task of QoS prediction. The performance of the forecasting method is compared from two viewpoints - the forecasting accuracy and the time cost.

The forecasting methods are evaluated on the real-world dataset of QoS attributes that was captured using a centralized passive QoS measurement technique. The results of the evaluation reveal that LSTM NN gives improved forecasting accuracy for the majority of the QoS attributes with higher granularity compared to other methods. The time cost of the LSTM NN is comparable with ARIMA model. To make our research reproducible, we make our comparison, including datasets, available for the public at [7].

Further, we discuss the possibilities of the improvement of LSTM NN performance by altering the used optimizer. We also inspect the influence of the choice of initial weights of LSTM NN to the forecast accuracy and discussed the characteristics of metrics used for performance comparison. The optimization of the LSTM NN performance and the k-step prediction for global web services QoS are interesting topics for the future research.

Acknowledgments

The data collection and experiment technical implementation and evaluation were supported by the Technology Agency of the Czech Republic under No. TA04010062 "Research and Development of Advanced Analytics Tools for Security and Performance Analysis of Network Infrastructure, Applications and Services". The methodology and results discussion was supported by "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822) supported by ERDF.

References

[1] Y. Syu, S. P. Ma, J. Y. Kuo, and Y. Y. FanJiang, "A survey on automated service composition methods and related techniques," in *Proceedings - 2012 IEEE 9th International Conference on Services Computing, SCC 2012*. IEEE, jun 2012, pp. 290–297.

[2] Y. Syu, J.-Y. Kuo, and Y.-Y. Fanjiang, "Time series forecasting for dynamic quality of web services: An empirical study," *Journal of Systems and Software*, vol. 134, pp. 279–303, dec 2017.

[3] J. Gozdecki, A. Jajszczyk, and R. Stankiewicz, "Quality of service terminology in IP networks," *IEEE Communications Magazine*, vol. 41, no. 3, pp. 153–159, mar 2003.

[4] A. Amin, A. Colman, and L. Grunske, "An Approach to Forecasting QoS Attributes of Web Services Based on ARIMA and GARCH Models," in *2012 IEEE 19th International Conference on Web Services*. IEEE, jun 2012, pp. 74–81.

[5] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2911–2924, oct 2017.

[6] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997.

[7] T. Jirsik, S. Trcka, and P. Celeda, "Centralized Quality of Service Forecasting withLSTM Neural Networks: Dataset," 2018. [Online]. Available: https://github.com/CSIRT-MU/QoSForecastLSTM

[8] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[9] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," sep 2013.

[10] J. Li and A. W. Moore, "Forecasting Web Page Views: Methods and Observations," *Journal of Machine Learning Research*, vol. 9, pp. 2217–2250, 2008.

[11] G. E. P. Box and G. M. Jenkins, *Time Series Analysis : Forecasting and Control*. San Francisco, USA: Holden-Day Series in Time Series Analysis, Revised ed., 1976.

[12] C. Chatfield, "The Holt-Winters Forecasting Procedure," *Journal of the Royal Statistical Society*, vol. 27, no. 3, pp. 264–279, 1978.

[13] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[14] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, mar 1994.

[15] C. Olah, "Understanding LSTM Networks," 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[16] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," p. 23, sep 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[17] S. Ranger, "iPhone, AI and big data: Here's how Apple plans to protect your privacy," 2016. [Online]. Available: https://www.zdnet.com/article/ai-big-data-and-the-iphone-heres-how-apple-plans-to-protect-your-privacy/

[18] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet Traffic Forecasting using Neural Networks," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006, pp. 2635–2642.

[19] P. Newbold, J. C. Hoff, and W. Vandaele, "A Practical Guide to Box-Jenkins Forecasting." *Journal of the American Statistical Association*, vol. 79, no. 388, p. 943, dec 1984.

[20] J. Brownlee, "How to Improve Deep Learning Performance," pp. 1–28, 2016. [Online]. Available: http://machinelearningmastery.com/improve-deep-learning-performance/

[21] B. Basnet, "LSTM Optimizer Choice ?" 2016. [Online]. Available: https://deepdatascience.wordpress.com/2016/11/18/which-lstm-optimizer-to-use/

[22] B. Cavallo, M. Di Penta, and G. Canfora, "An empirical comparison of methods to support QoS-aware service selection," in *Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems - PESOS '10*. New York, New York, USA: ACM Press, 2010, p. 64.

[23] Y. Syu, C.-M. Wang, and Y.-Y. Fanjiang, "A Survey of Time-Aware Dynamic QoS Forecasting Research, Its Future Challenges and Research Directions," in *Internationa Conference on Services Computing SCC 2018*. Springer, Cham, jun 2018, pp. 36–50.

[24] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, and M. R. Lyu, "A Spatial-Temporal QoS Prediction Approach for Time-aware Web Service Recommendation," *ACM Transactions on the Web*, vol. 10, no. 1, pp. 1–25, feb 2016.

[25] A. Amin, L. Grunske, and A. Colman, "An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering - ASE 2012*. New York, New York, USA: ACM Press, 2012, p. 130.

[26] T. Senivongse and N. Wongsawangpanich, "Composing Services of Different Granularity and Varying QoS Us-

ing Genetic Algorithm," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. I, San Francisco, USA, 2011, p. 6.

[27] M. H. Zadeh and M. A. Seyyedi, "Qos monitoring for web services by Time Series Forecasting," in *Proceedings - 2010 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010*, vol. 5, 2010, pp. 659–663.

[28] J. Bendriss, I. G. B. Yahia, P. Chemouil, and D. Zeghlache, "AI for SLA Management in Programmable Networks," in *DRCN 2017 - Design of Reliable Communication Networks*. Munich, Germany: VDE, 2017, p. 8.