MSP-RNN: Multi-Step Piecewise Recurrent Neural Network for Predicting the Tendency of Services Invocation

Haozhe Lin, Yushun Fan, Jia Zhang, *Senior Member, IEEE,* Bing Bai

Abstract—Driven by the widespread application of Service-Oriented Architecture (SOA), an increasing number of services and mashups have been developed and published onto the Internet in the past decades. With the number keeping on burgeoning, predicting the tendency of services invocation will provide various roles in service ecosystems with promising opportunities. However, services invocation bear three unique characteristics, which give rise to difficulties in predicting them. Firstly, enormous services show different and complicated traits, like periodicity, nonlinearity and nonstationarity. Secondly, services providing similar or compensatory functions make up intricate relationship. Thirdly, the combination dependencies between mashups and their comprising component services further amplify the difficulty. Given these factors, we have developed a tailored model Multi-Step Piecewise Recurrent Neural Network (MSP-RNN) to predict the tendency of services invocation. In MSP-RNN, Long Short Term Memory (LSTM) units are used to extract universal features. Based on these features, we have developed a piecewise regressive mechanism to make prediction discriminatingly. Besides, we have developed a multi-step prediction strategy to further enhance prediction accuracy and robustness. Extensive experiments in real-world data set with interpretable analysis show that MSP-RNN predicts the tendency of services invocation more accurately, i.e., by 3.7% in terms of symmetric mean absolute percentage error (SMAPE), than state-of-the-art baseline methods.

Index Terms—Service Discovery, Tendency of Services Invocation, Deep Learning, Time Series Prediction.

1 Introduction

With the wide application of Service-Oriented Architecture (SOA) and Cloud Computing [1], a burgeoning number of services have been developed and published into service ecosystems *(e.g., ProgrammableWeb.com*[1]*)* in the past decades. Based on the enormous available services, *i.e., APIs*, software developers have been leveraging them as reusable components to create value-added new products, *i.e., service compositions or mashups* [2], to meet various user requirements. Such a service-oriented software engineering methodology has changed the business mode and in turn made service ecosystems progressively prosperous [3]. Under this context, predicting the tendency of services invocation, *i.e., the number of services invocation at a given future time period*, is becoming increasingly important. Such a prediction may benefit service providers, service users, as well as system administrators. For example, service providers can exploit the foreseen traffic to dynamically adjust their computational resources rather than applying a static strategy, so that they can operate their services in a more economic way. For service users, accurate prediction will allow them to choose the optimal services that better satisfy their requirements [4]. Meanwhile, system administrators can also utilize the prediction to alert service developers or provide suggestions in advance [5].

Regarding services invocation tendency prediction, one may associate with time series forecasting technique, like weather forecast. However, predicting the tendency of services invocation bears three unique features to which have to be paid special attention. (i) **Complicated Traits**. In a service ecosystem, each service shows complicated traits in periodicity, nonlinearity and nonstationarity. Moreover, the attributes of these enormous services even differ from each other. Take the service *World Cup in JSON* as an example. For it provides users with up-to-date information about World Cup progress, it tends to be intensively invoked while the World Cup is being held. Thus it shows a four-year periodicity without distinct patterns in other days, which is apparently different from many traits of other services. (ii) **Intricate Relationship**. In our previous work [6], we have found that there exists intricate relationship among services due to their similar or compensatory functions, and we believe that such a relationship is useful for predicting the tendency of services invocation. For example, *Last.fm* and *Spotify* are two services both supplying streaming media service. Therefore, when a famous singer releases a new album, we would be more confident that *Spotify* would witness a high invocation, if a prominent invocation tendency of *Last.fm* could be observed. (iii) **Combination Dependency**. Since mashups are developed based on existing services, the invocation times of one service will not only be ascribed to itself, but also result from its related mashups.

- *H. Lin, Y. Fan are with the Department of Automation, Tsinghua University, Beijing, China.*
  E-mail: linhz16@mails.tsinghua.edu.cn, fanyus@tsinghua.edu.cn.
- *J. Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, CA.*
  E-mail: jia.zhang@sv.cmu.edu.
- *B. Bai is with the Cloud and Smart Industries Group, Tencent, Beijing, China.*
  E-mail: icebai@tencent.com.

Manuscript received June 5, 2019.

1. https://www.programmableweb.com

In other words, once a mashup is invoked, all component services will be invoked at the same time, even though they are not similar in their functions or descriptions at all. For instance, *Foodsta* is a mashup with *Instagram* and *Google Maps* as its component services. When *Foodsta* is invoked, both *Instagram* and *Google Maps* will be invoked as well. Consequently, these intractable facts remarkably discriminate the service invocation prediction problem from other time-series prediction problems.

Given these unique features, no existing time series predicting methods could fully address the problem concerning services invocation prediction. Traditional univariate methods, such as Autoregressive Integrated Moving Average model (ARIMA) [7] and Support Vector Regression (SVR) [8], focus on predicting an individual time series. For such methods, the tendency of one time series will be predicted based on its past observation, which can be regarded as targeted prediction or discriminating prediction. Apparently, such methods can only focus on intrinsic traits of one sequence, but waste the potential of the intricate relationship among sequences. In recent years, with in-depth research about deep learning, Recurrent Neural Networks (RNN) [9] have become auspicious approaches to predicting time series. For this model, sequences will be treated as a whole and universal changing patterns among sequences will be learned. However, comparing to univariate time series prediction models, RNN making prediction generally impairs the pertinence of prediction, and thus shows a limitation.

Intuitively, taking into consideration of both generality and pertinence of prediction may be helpful to predict the tendency of services invocation more accurately. In this context, we have developed a novel deep neural network, named Multi-Step Piecewise Recurrent Neural Network (MSP-RNN), for predicting the tendency of services invocation. As for the model structure, MSP-RNN first leverages the Long Short Term Memory (LSTM) [10] to capture general traits of service invocation sequences, such as periodicity, nonlinearity and long-term dependencies. Afterwards, MSP-RNN develops a piecewise regressive mechanism to stimulate the pertinence of prediction. Through the piecewise regressive mechanism, on the one hand, MSP-RNN clusters service invocation sequences into a collection of categories; and on the other hand, MSP-RNN trains parallel fully connected layers for different categories to regress corresponding intermediate results. During training process, MSP-RNN will continuously adjust the cluster mode and regression way until the optimal equilibrium is achieved. As a result, the tendency of services invocation will be predicted discriminatingly. As for parameters learning, we have designed a subtle loss function with several regularization techniques to ensure the efficacy of the piecewise regressive mechanism. We have also developed a multi-step prediction strategy to further enhance the prediction accuracy, as well as to improve the robustness of our model. Such a strategy enables our MSP-RNN to predict more accurately than baseline methods in a longer time range. To our best knowledge, this work is the first effort to apply deep learning techniques to predict the tendency of service invocation, considering both generality and pertinence.

To test and verify the performance of our MSP-RNN against its main opponents, we designed and conducted a collection of experiments on real-world data set. The experimental results indicate that MSP-RNN remarkably outperforms baseline models. By visualizing the inner parts of MSP-RNN, it can be proved that the improvements of prediction accuracy brought by MSP-RNN are interpretable.

By incorporating the aforementioned points, the main contributions of this work are summarized as follows:

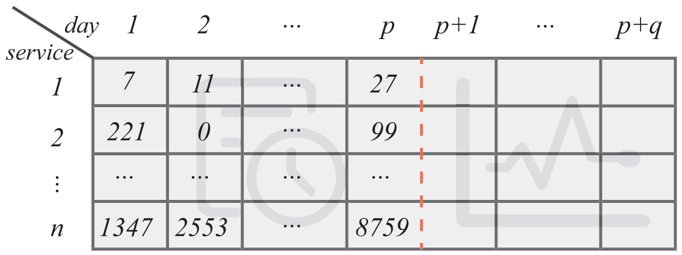
- We have proposed a piecewise regressive mechanism. Integrating with LSTM, our model can predict the tendency of services invocation, by taking into consideration of both generality and pertinence.
- We have developed a multi-step prediction strategy. Through this strategy, prediction accuracy could be further enhanced, and the robustness of MSP-RNN could also be increased.
- We designed and conducted a collection of experiments on real-world data set, which show that MSP-RNN outperforms baseline methods in terms of prediction accuracy, especially when prediction periods are longer.

The rest of the paper is organized as follows. Section 2 restates our problem mathematically. Section 3 describes the framework of our MSP-RNN model. Section 4 introduces the details of parameter learning. Section 5 reports the experimental settings, results and analysis. Section 6 discusses the related work. Finally, Section 7 draws conclusions.

2 Notations and Problem Definition

In this section, we will first introduce notations, and then formulate the problem mathematically.

Definition 1 (Service invocation records). Given one service, its invocation records refer to its invocation times in the past p days. We denote by a time series $\boldsymbol{x}_i$ the service invocation records, where $\boldsymbol{x}_i$ can be broken down to $\boldsymbol{x}_i = \{x_{i,0}, x_{i,1}, \dots, x_{i,p}\}$ for service i in the past p days. More specifically, $x_{i,t}$ $(t \in [0, p])$ represents the number of invocation times for service i in the the past t-th day. In this paper, we regard the number of services n in a service ecosystem significant, 20,000 at least.



service \ day	1	2	···	p	p+1	···	p+q
1	7	11	···	27			
2	221	0	···	99			
⋮	···	···	···	···			
n	1347	2553	···	8759			

Fig. 1: Predicting the tendency of services invocation. The aim is to fill out the missing values in this table as tendency of services invocation *(i.e., values after the red dashed line)* for n services in a service ecosystem, based on service invocation records we have already known in the past p days.

Definition 2 (Tendency of service invocation). The tendency of one service invocation refers to the invocation

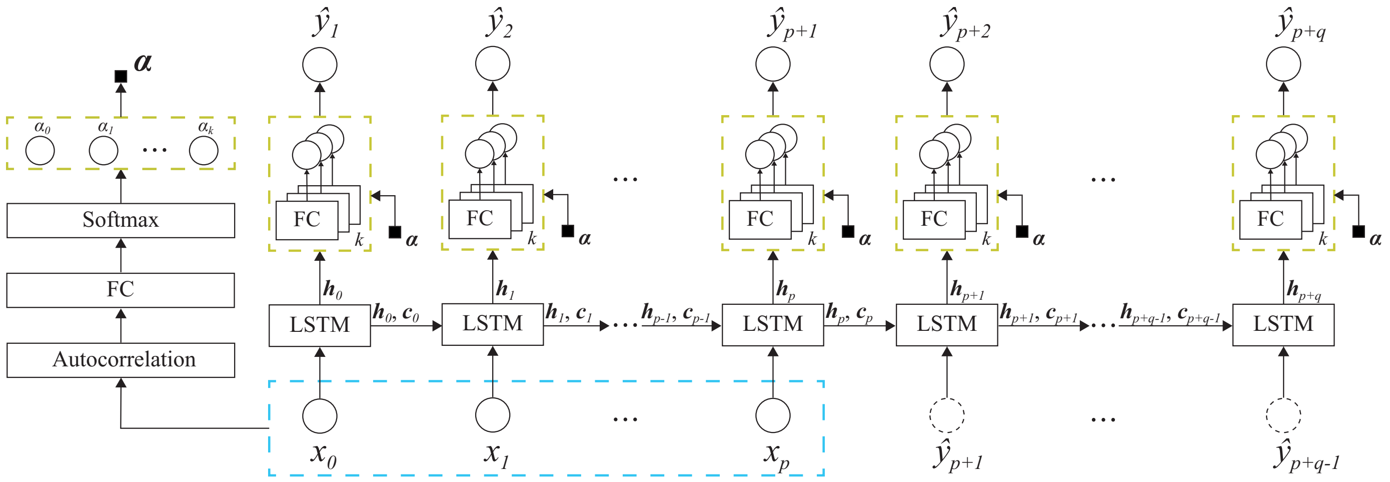


Fig. 2: Network structure of MSP-RNN. Being a layered structure, MSP-RNN takes a batch of service invocation records $\boldsymbol{x}$ as inputs, which are shown in the blue dashed rectangle. These records will be utilized in two different ways, one being information source of an LSTM-based feature extraction module, the other providing autocorrelation coefficients for a piecewise regressive mechanism. At the lower layer, MSP-RNN applies LSTM units to extract general features of sequences, and to represent them as hidden states $\boldsymbol{h}$ in every time step. At the upper layer, in the yellow dashed rectangle, MSP-RNN develops a piecewise regressive mechanism to predict the tendency of service invocation $\hat{y}_t$ discriminatingly, where several FCs refer to fully connected layers. During training, MSP-RNN applies a multi-step prediction strategy to forecast invocation tendency in the subsequent several steps, and uses such results to update model parameters. During forecasting, MSP-RNN treats its previous prediction values as the inputs of the following steps to predict the tendency of services invocation.

times of the service in the next q days. We denote by $\hat{\boldsymbol{y}}_i = \{\hat{y}_{i,p+1}, \hat{y}_{i,p+2}, \ldots, \hat{y}_{i,p+q}\}$ the tendency of service invocation for service i in the next q days, where $\hat{y}_{i,t}$ $(t \in (p, p+q])$ represents the predicted value of invocation times for service i in the next t-th day.

Based on the formal definitions, Fig. 1 illustrates problem 1 definition vividly.

Problem 1 (Predicting the tendency of services invocation). Given n services in a service ecosystem, we define the problem as predicting the tendency of services invocation for all n services in the next q days *(i.e.,* $\hat{\boldsymbol{y}}_i = \{\hat{y}_{i,p+1}, \hat{y}_{i,p+2}, \ldots, \hat{y}_{i,p+q}\}$ *for all i)*, by making use of the services invocation records made known in the past p days *(i.e.,* $\boldsymbol{x}_i = \{x_{i,0}, x_{i,1}, \ldots, x_{i,p}\}$ *for all i).*

3 Model Framework

In this section, we will elaborate our solution Multi-Step Piecewise Recurrent Neural Network (MSP-RNN) for predicting the tendency of services invocation, which comprises two main parts: an LSTM-based feature extraction module and a piecewise regressive mechanism. (A multi-step prediction strategy will be introduced in Section 4.) Fig. 2 shows its network structure.

3.1 LSTM-based Feature Extraction Module

To predict the tendency of services invocation, the changing patterns of sequences could be modeled according to the past records. With the in-depth study of Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), a special structure of RNN, has been proved to be capable of modeling some meaningful states of sequences [11]. For example, in text modeling tasks, LSTM is able to keep track of line lengths, quotes and brackets. Similarly, in our problem, we believe that LSTM units can capture the complicated intrinsic traits among service invocation sequences, since the tendency of services invocation shows strong correlation with their past records.

As shown in Fig. 3, an LSTM unit is composed of a memory cell and three gates, i.e., forget gate, input gate and output gate, enabling modification of the cell memory, whose functions can be defined as Eq. 1.

$$\begin{aligned}
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1}, x_t] + \boldsymbol{b}_f) \\
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i[\boldsymbol{h}_{t-1}, x_t] + \boldsymbol{b}_i) \\
\widetilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}_c[\boldsymbol{h}_{t-1}, x_t] + \boldsymbol{b}_c) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \widetilde{\boldsymbol{c}}_t \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, x_t] + \boldsymbol{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned} \tag{1}$$

where $\odot$ represents the element-wise product; $\boldsymbol{W}_f$, $\boldsymbol{W}_i$, $\boldsymbol{W}_o$ and $\boldsymbol{W}_c$ represent the parameters of the forget gate, input gate, output gate and the memory cell, respectively; $\sigma(\cdot)$ and $\tanh(\cdot)$ represent sigmoid function and hyperbolic tangent function, respectively.

Once receiving sufficient training, LSTM will project these records into a high-dimension space, where each dimension presents a meaningful state, probably being different kinds of periodicity or other features. Based on these effective features, i.e., the hidden states from LSTM, a fully connected layer (FC as shown in Fig. 3) will be able to transform them into prediction values, and present an acceptable prediction accuracy [12]. In other words, because

LSTM takes all service invocation records as a whole, the generality of prediction is conserved. However, since the changing patterns of service invocation sequences fairly differ from each other, the pertinence of prediction demands further consideration.

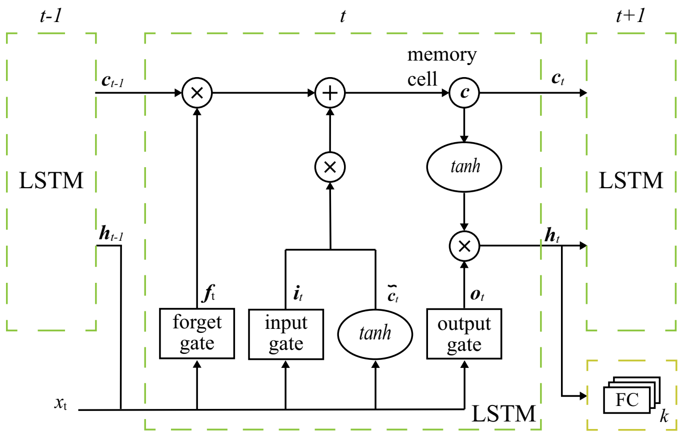


Fig. 3: LSTM-based feature extracture module. Memory cells $\boldsymbol{c}$ in different units record historical information with different meanings, and three gates, *i.e., forget gate, input gate and output gate*, execute sigmoid functions to modify the memory at every time step. The hidden states will be used as input of both LSTM units in next step and parallel fully connected layers (FCs).

3.2 Piecewise Regressive Mechanism

To further improve prediction accuracy, we make a reasonable assumption that different kinds of service invocation sequences will pay different attention to different hidden states extracted by LSTM. So we expect that training multiple parallel fully connected layers will induce MSP-RNN to predict the tendency of services invocation differently. Thus, we propose a piecewise regressive mechanism to make prediction discriminatingly, which idea is depicted in Fig. 4.

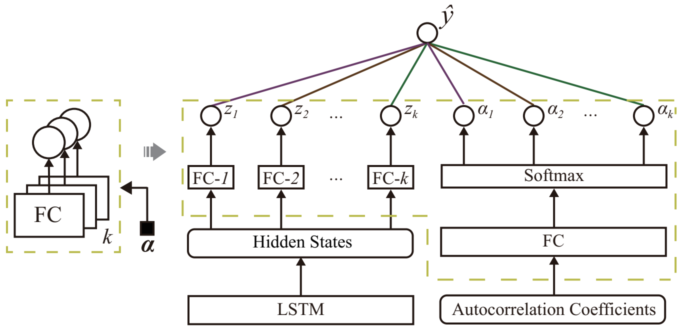


Fig. 4: Overview of Piecewise regressive mechanism.

The piecewise regressive mechanism comprises two functions. On the one hand, it clusters service invocation sequences into k categories with class probabilities $\boldsymbol{\alpha}$. On the other hand, it regresses the corresponding intermediate results z_k by several parallel fully connected layers (FCs). The combination of the results of these two functions generates the final prediction value $\hat{y}$.

The major process of the piecewise regressive mechanism is illustrated in the yellow dotted polygon. As shown in Fig. 4, the prerequisite of applying the piecewise regressive mechanism is to determine a regression method and a clustering pattern. With respect to the regression method, we take hidden states from LSTM (Eq. 1) as effective features and apply parallel fully connected layers to synthesize them as intermediate results, which can be described by Eq. 2.

$$z_k = \boldsymbol{w}_k^\top \cdot \boldsymbol{h} + \boldsymbol{b} \tag{2}$$

where various $\boldsymbol{w}_k$ refer to weight parameters of different fully connected layers, $\boldsymbol{h}$ refers to the hidden states extracted by LSTM units, $\boldsymbol{b}$ refers to the bias of this layer, and various z_k refer to the intermediate results. It should be noted that the piecewise regressive mechanism, combining several intermediate results to generate finial prediction, is slightly similar with the maxout active function in modeling nonlinearity [13]. Therefore, it is fine to place an active function after each individual fully connected layer. However, given computational complexity associated, we remove the active function to accelerate the training process.

To cluster service invocation sequences into different categories, it might be tricky to utilize raw dates directly, because the underneath rule of service invocation records is subtle and the length of the time series is endless [14]. After many trials, we discovered that autocorrelation coefficients reflecting correlation of one sequence can be influential features. Consequently, we adopt the autocorrelation function in MSP-RNN, to transform the original sequential data into another way, which is formulated in Eq. 3. Based on the autocorrelation coefficients, we then set up a softmax classifier,i.e., the softmax function together with the FC between it and autocorrelation, to obtain various class probabilities α_k in Eq. 4.

$$\rho_\tau = \frac{\mathbb{E}\left[(x_t - \mu)(x_{t+\tau} - \mu)\right]}{\sigma^2} \tag{3}$$

$$\boldsymbol{\alpha} = \frac{e^{\boldsymbol{w}_s^\top \cdot \boldsymbol{\rho}}}{\sum_{k=1}^{n_{\text{class}}} e^{\boldsymbol{w}_s^\top \cdot \boldsymbol{\rho}}} \tag{4}$$

where ρ_τ represents the autocorrelation coefficient with time lag τ, $\mathbb{E}$ represents the expected value operator, μ represents the mean value of one service invocation sequence, σ^2 represents the variance of one service invocation sequence, x_t represents the invocation records of one service in day t, $\boldsymbol{w}_s$ represents the parameters of the fully connected layer between the autocorrelation function and the softmax function, $\boldsymbol{\alpha}$ represents the class probabilities that can be broken down to $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, \ldots, \alpha_k\}$ to represent the probability of one service invocation sequence belonging to class k, and n_{class} represents the number of the clusters predefined.

To boost the piecewise regressive mechanism, we further develop a threshold-based smoothing technique over the class probabilities outputted by the network. In detail, we smooth insignificant α into very small values (Eq. 5), and then re-scale them into a probabilistic scale (Eq. 6). As a consequence, the impact of the specific fully connected layers is intensified, so as to ensure to apply proper regression strategies over different clusters. By comparison, if we directly apply original probabilities without re-scaling, the neural network will probably tend to train several similar fully connected layers.

$$\alpha_i = \begin{cases} \alpha_i, & \text{if} \quad \alpha_i \leq \frac{1}{n_{\text{class}}} \\ 0.01, & \text{otherwise} \end{cases} \tag{5}$$

$$\alpha_i = \frac{\alpha_i}{\sum_k \alpha_k} \tag{6}$$

Ultimately, as shown in Fig. 4, at the end of each forward propagating process, through the piecewise regressive mechanism, MSP-RNN will integrate all these intermediate results $\boldsymbol{z}$ and their corresponding class probabilities $\boldsymbol{\alpha}$ to produce the final prediction $\hat{y}_t$, which is shown in Eq. 7.

$$\hat{y} = \sum_{k=1}^{n_{\text{class}}} \alpha_k \cdot z_k \tag{7}$$

In theory, the piecewise regressive mechanism shall not lead to a worse prediction result, comparing to the vanilla RNN without it. Assume we meet an egregious situation where all the service invocation sequences are clustered into the same category, which implies that only one fully connected layer will be activated while others will be deactivated. In such an extreme situation, MSP-RNN will degenerate into a vanilla RNN. In other situations, when the piecewise regressive mechanism works, the tendency of services invocation will not only be predicted generally but also be predicted discriminatingly based on corresponding categories. Actually, based on the piecewise regressive mechanism, during backward propagation, the cluster mode of a regressive way will be ceaselessly adjusted until the best equilibrium is reached. As a result, the service invocation sequences will be predicted by an ensemble model, which will eventually give rise to the enhancement of prediction accuracy.

4 Parameters Learning

In this section, we carefully analyze the aspects regarding the prediction effect and training efficiency of MSP-RNN.

4.1 Loss Function

The piecewise regressive mechanism sounds inspiring, but what if all the fully connected layers happen to decode hidden states by the same way? Our experiments found that only an ingenious loss function can ensure the mechanism functioning properly. Otherwise, the piecewise regressive mechanism, *i.e., the structure shown in Fig. 4*, sometimes may even lead to several identical fully connected layers, performing just like a vanilla RNN. For this sake, we have designed a dedicated loss function for MSP-RNN from three perspectives, which is devised as Eq. 8:

$$\begin{aligned}\mathcal{L} =& \frac{1}{n}\sum_{(i,t)}\left|\log\frac{y_{it}+1}{\hat{y}_{it}+1}\right| + \lambda\frac{\boldsymbol{w}_p \cdot \boldsymbol{w}_q}{\parallel \boldsymbol{w}_p \parallel_2 \cdot \parallel \boldsymbol{w}_q \parallel_2} + \\ & \sum_k \gamma_k \parallel \boldsymbol{w}_k \parallel_1\end{aligned} \tag{8}$$

where n represents the number of samples in a batch, $\hat{y}_{it}$ represents the prediction value of MSP-RNN for service i in day t, y_{it} represents the true invocation records for service i in day t, various γ and λ represent hyper-parameters determining the intensity of regularization, respectively. Various $\boldsymbol{w}$ in the above function represent the weight parameters of parallel fully connected layers just after the LSTM layer, where $\boldsymbol{w}_p$ and $\boldsymbol{w}_q$ represent the wight parameters of two of them randomly selected.

The first part of the loss function reflects the discrepancy between prediction values and real values. In many time series prediction scenarios, people tend to measure the discrepancy by some absolute indicators, like mean square error (MSE), mean absolute error (MAE), and so on. However, these indicators are inappreciable here in our case. In this problem of service invocation prediction, service invocation records considerably vary in the orders of magnitude. If such absolute indicators are applied, it will lead to a situation where a service invocation sequence with a larger order of magnitude will impact model parameters much more than a sequence with a smaller one. As a result, the model will fit mainly according to the sequences with larger orders of magnitude, while neglecting the smaller ones. Therefore, we choose to evaluate our model in a logarithmic scale, which will eliminate the influence caused by the order of magnitude to some extent. Besides, being a relative indicator, the first item of our loss function makes it more meaningful than an absolute one.

The second part and third part of the loss function act as a catalyst for the piecewise regressive mechanism, which also enhance the interpretability of our MSP-RNN. Since we hope that multiple fully connected layers can be learned discriminatingly for service invocation sequences with different traits, we first measure cosine similarity between two fully connected layers as a part of loss. On the one hand, we allocate λ with a very slight value, so at the initial training stage, the decoding way of fully connected layers will be discriminated directly. On the other hand, we set a larger predefined cluster number to allow redundancy. As mentioned before, if only one fully connected layer conducts the decoding process while others are smoothed to near zero, MSP-RNN will degenerate as a vanilla RNN. Such a setting will ensure that the prediction accuracy will not become too bad under this extreme situation. Under other circumstances, the piecewise regressive mechanism will bring about promising results. Finally, the third item is an ℓ_1 norm regularization, which can lead to sparse solutions of multiple fully connected layers $\boldsymbol{w}_k$, implying that different fully connected layers pay different attentions to different hidden states. As a consequence, the piecewise regressive mechanism will avoid service invocation sequences from being clustered into one category, and increase prediction accuracy eventually.

4.2 Multi-Step Prediction Strategy

After elaborating the piecewise regressive mechanism, we have developed a multi-step prediction strategy to take place of the single-step prediction strategy in our previous work [15]. This is because in time series prediction tasks, RNNs are inclined to assign great importance to the input in current time step, *i.e., service invocation record x_t in our problem*. This implies that trained models would value the

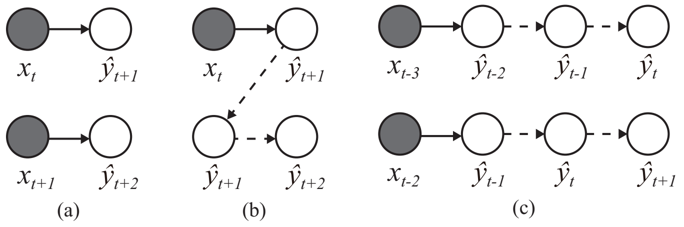


Fig. 5: Training and predicting processes of PRNN and MSP-RNN. (a) and (c) represent the training process of PRNN and MSP-RNN, respectively; (b) represents the predicting process of both models.

hidden states *(e.g.,* $\boldsymbol{h}_{t-1}$ *from LSTM)* less, even though they might have contained useful meanings. Typically, single-step prediction strategy is very useful if only few forward values are required to be forecast. However, since we expect to estimate the invocation tendency of services in a longer time period, (e.g., 2 months at least), a multi-step prediction strategy seems more appropriate, which can be illustrated in Fig. 5.

When training a model for PRNN, a single-step predicting strategy is adopted. As shown in Fig. 5 (a), we use the observed value x_t at day t to predict the invocation times in the next day *(i.e.,* $\hat{y}_{t+1}$*)*, and use the observed value x_{t+1} at day $(t+1)$ to predict the invocation times in the following day *(i.e.,* $\hat{y}_{t+2}$*)*. Train loss will be calculated by only the differences between each pair. By contrast, for MSP-RNN, we extend the strategy by using the observed value to predict the invocation times in the next few days in an autoregressive manner, (e.g., use x_{t-3} to predict for day $\hat{y}_{t-2}$, $\hat{y}_{t-1}$, and $\hat{y}_t$) as shown in Fig. 5 (c). All the prediction results $\{\hat{y}_t,\ t \leq n_{\text{step}}\}$ will be used to back propagate gradients. Under this context, it it natural that MSP-RNN has to count more on previous hidden states, and the expression ability of MSP-RNN will be in turn enhanced. Nevertheless when predicting, it is certainly better to use single-step strategy to make prediction, because the information x_t at the current moment is definitely more valuable than a predicted one, since the model has already known how to utilize its hidden states.

5 EXPERIMENTS

In this section, we will report a collection of experiments designed on real-world data set to test and verify the performance of our MSP-RNN.

5.1 Data Set

Since the real service invocation records are inadequate, we conducted experiments on a public data set Wikistat[2], which records millions of Web Page Views (PV) from English, Chinese and other wiki-projects, hourly since September 2013. The reason why we adopted such a substitution is due to three aspects. Firstly, Web page views bear complex characteristics, which are comparable to those of services invocation. For example, a Wikipage *World Cup*, because of its content, presents similar traits of service *World Cup in JSON* we mentioned in the introduction section. Meanwhile, there are also millions of Web pages hardly being viewed by people, which is similar with the traits of long-tail services in a service ecosystem [16]. Secondly, there are also intricate relationships among Web pages. One example is that when people visit a Web page about a character in Marvel movies, like *Iron Man*[3] in Wikipedia, it is likely that the Wikipage about *Captain America*[4] will be visited as well. Thirdly, it seems that combination dependencies also exist in this context. Analogous to services that may provide several functions or operations, we regard wikipages as services and the hyperlinks in a wikipage as its functions or operations in our experiments. Similar with mashups in a service ecosystem, we deem a new composition is created, if two or more Web pages are always viewed in a very short time period. Given these reasons, we believe that it is reasonable to take Web pages as analog of services and regard the number of access to a Web page as the invocation number of a service.

In our experiments, we randomly picked up 29,839 pages from English and Chinese projects, and divided them into two parts. The first part, starting from July 1, 2015 to June 30, 2017 and containing 731 points for each, were used to train and validate the models. The rest one, starting from July 1, 2017 to August 31, 2017 and containing 62 points for each, were used to evaluate the prediction accuracy and robustness of our model. Numerical properties of the data set are summarized in Table 1 (PV refers to page view).

TABLE 1: Numerical properties of dataset

Item	Number
PVs with decadal order of magnitudes	6,331
PVs with hundred order of magnitudes	10,567
PVs with thousand order of magnitudes	8,996
PVs with other order of magnitudes	3,945
samples of PVs for training	2.1×10^7
samples of PVs for testing	1.8×10^6

5.2 Evaluation Scheme

To evaluate our model, we adopted two indicators, Symmetric Mean Absolute Percentage Error (SMAPE) [17] and Dynamic Time Warping (DTW) [18].

The first indicator SMAPE is formulated by Eq. 9:

$$\text{SMAPE} = \frac{1}{n}\sum_t \frac{|\hat{\boldsymbol{Y}}_t - \boldsymbol{Y}_t|}{(\hat{\boldsymbol{Y}}_t + \boldsymbol{Y}_t)/2} \tag{9}$$

where n represents the number of samples in a training batch, $\hat{\boldsymbol{Y}}_t$ represents the prediction values of the invocation times for all services (or PVs in the experiments), and $\boldsymbol{Y}_t$ represents the real values. It is obvious that a lower SMAPE value means a higher prediction accuracy.

As shown in Table 1, page views fairly vary in the orders of magnitude, which make it inappropriate to be evaluated by the most common indicators, like Mean Square Error (MSE) and Mean Absolute Error (MAE), etc. Imaging there are ten sequences, one is in thousand scale, while

2. https://dumps.wikimedia.org

3. https://en.wikipedia.org/wiki/Iron_Man
4. https://en.wikipedia.org/wiki/Captain_America

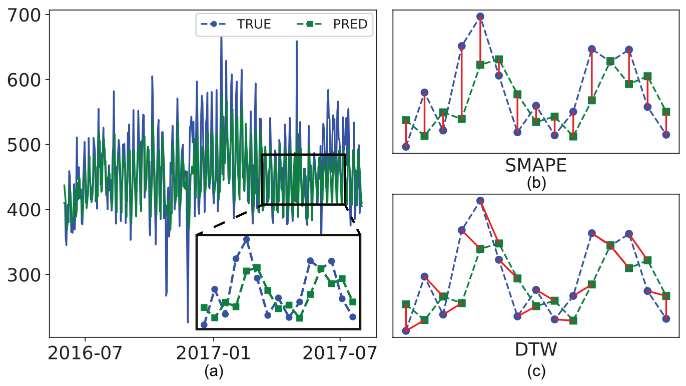


Fig. 6: Two evaluation schemes. (a) depicts the observed values and predicted values of one service invocation sequence. (b) and (c) refer to different ways to measure prediction accuracy, with red lines as their alignment. For SMAPE, two trajectories will be measured by calculating pointwise residual, while for DTW, two trajectories will be measured by calculating residual of optimal alignment.

others are in decadal scale. When the one in thousand scale is predicted inaccurately, no matter how precisely the others are predicted, a model will be deemed as an awful one, if MSE or MAE are used to evaluate. By comparison, SMAPE evaluates the relative error between true values and predicted values, which is proper in this task, because it will not be affected by the order of magnitude. Besides, in terms of SMAPE, under-forecasting prediction gets higher value than over-forecasting one. It is quite suitable for our problem, because slight redundancy is essential for any kinds of roles in service ecosystem as explained in earlier sections.

The second indicator DTW is a shape-based similarity measurement, widely used in speech recognition, time bargain and some other time series inference and prediction problems. A lower DTW stands for a higher shape similarity between two sequences. Different from other indicators, DTW places an extra emphasis on similarity of trajectory between two sequences. Take Fig. 6 as an example, where the blue line and the green line are observed invocation records and predicted invocation tendency of one service, respectively. For SMAPE, the prediction accuracy will be evaluated according to pointwise residual. For DTW, the accuracy will be evaluated through measuring the similarity of trajectory. More specifically, DTW will find the optimal alignment between two sequences, and then calculate corresponding residual.

5.3 Benchmarks and Hyper-parameters Setting

To test and verify the performance of our MSP-RNN, we compared it with the following baseline methods.

- **ARIMA.** As one of the most classical time series prediction models, Autoregressive Integrated Moving Average model (ARIMA) was applied to treat each service invocation sequence as an individual to make prediction [7]. In our experiments, we applied Dickey-Fuller test [19] and Bayesian Information Criterion (BIC) to determine whether to apply Autoregressive Moving Average (ARMA) or ARIMA. We modeled stationary sequences through the ARMA model (Eq. 10) and nonstationary ones through the ARIMA model. For some PVs whose orders caused singular value decomposition exception in the experiments, we set the last day of training set (June 30, 2017) as the prediction results. We implemented this method through the *statsmodel* python package[5].

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{10}$$

where various φ_p and θ_i refer to parameters of ARMA, c refers to a constant, and ε_t refers to white noise.
- **SVR.** Support Vector Regression (SVR) has been widely used in predicting time series since 2004, because it presents an impressive ability to model nonlinearity of sequences [20], [21]. In our experiments, we treated each service invocation sequence as an individual, and set one's past 731 days as training set to model their invocation patterns. After training, we used each model to generate corresponding prediction values. For all sequences, we set the polynomial kernel for each SVR model, because it performed the best among all optional kernels when validating. We implemented this method through the *sklearn* python package[6].
- **LSTM-based RNN.** Recurrent neural networks have been acknowledged as a promising way to predict time series in recent years. Particularly, Long Short-Term Memory (LSTM), a special structure of RNNs, is capable of capturing long-term dependency of sequences, which has widely been used for predicting traffic, weather, stock price and some other domains [22], [23], [24]. In our experiments, we built an LSTM-based model treating all sequences equally to predict the tendency of services invocation.
- **LSTNet.** Long- and Short-term Time-series network is the state-of-the-art method to capture both long-term and short-term pattern of time series [25].

In the following experiments, all neural network-based models (including our MSP-RNN) shared the same batch size ($n_{\text{batch}} = 128$), hidden size in recurrent layer ($n_{\text{h}} = 200$), and they were optimized by Adam [26] with an initial learning rate as $lr = 0.001$. We set intensity factor $\gamma = 0.1$ for cosine similarity in loss function to discriminate parallel fully connected layers, and set intensity factor $\lambda = 0.01$ to each ℓ_1 regularization items to stimulate sparse solutions. For MSP-RNN, we predefined cluster number as $n_{\text{class}} = 5$, and step length as $n_{\text{step}} = 3$, which will be carefully discussed in the following sections.

5.4 Evaluation Results

5.4.1 Accuracy Comparison

Prediction accuracy is one of the most important properties needed to be discreetly evaluated. Therefore, we conducted 10 repeated experiments with the baselines methods as a

5. https://www.statsmodels.org
6. https://pypi.org/project/scikit-learn

comparison to verify the performance of our MSP-RNN model. Fig. 7 and Fig. 8 illustrate the SMAPE and DTW of MSP-RNN and other baseline methods with two months (62 days) prediction.

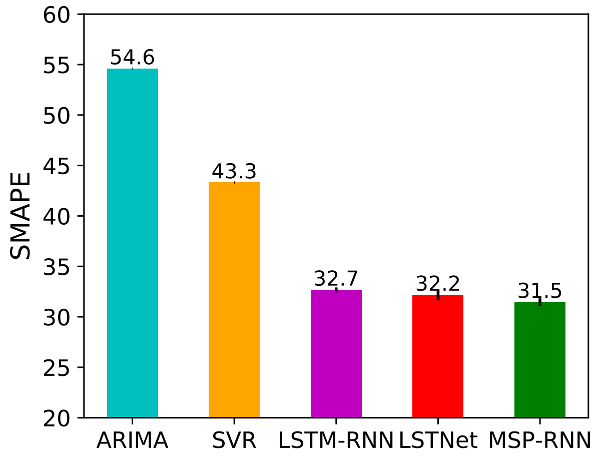


Fig. 7: SMAPE of ARIMA, SVR, LSTM-RNN and MSP-RNN.

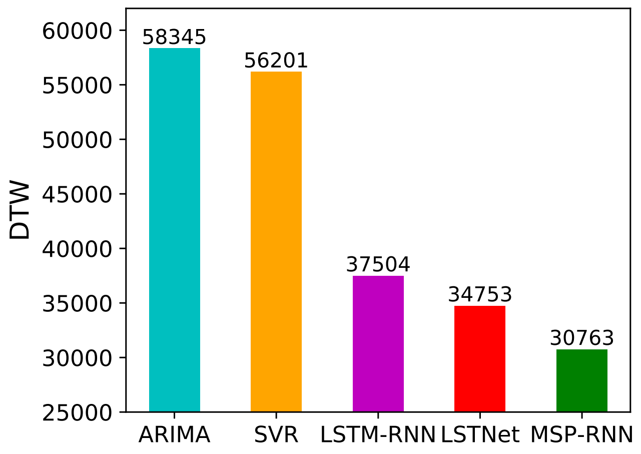


Fig. 8: DTW of ARIMA, SVR, LSTM-RNN and MSP-RNN.

In terms of SMAPE, the prediction error of MSP-RNN is prominently lower than those of ARIMA and SVR. Comparing to LSTM-RNN, MSP-RNN also decreases by 3.7%. In terms of DTW, MSP-RNN outperforms other methods more drastically, with 17.9% drop comparing to LSTM-RNN. Generally speaking, MSP-RNN is excellent among these methods. We thus ascribe the remarkable increment of the LSTM-based models to the LSTM component, which is able to extract efficient features of sequences. We also believe that the further increment of our MSP-RNN should be contributed to the piecewise regressive mechanism enabling the model to make prediction discriminatingly, and the multi-step prediction strategy enhancing the robustness of the model.

Since the LSTM-based models perform much better than the other methods, we further compare the performance of these methods with growing prediction days. Fig. 9 depicts SMAPE of LSTM-RNN, PRNN (a degeneration of MSP-RNN, which we elaborate in [15]), and MSP-RNN with growing prediction range. As shown in this figure, the performance of PRNN and MSP-RNN are always better than that of LSTM-RNN, especially in the first few prediction spans. The phenomenon justifies that the piecewise regressive mechanism is effective. Besides, with the prediction span keeps on growing, MSP-RNN predicts more accurately than PRNN, although they are neck-and-neck in very initial prediction span. This indicates that the multi-step prediction strategy is helpful for improving the robustness of our model.

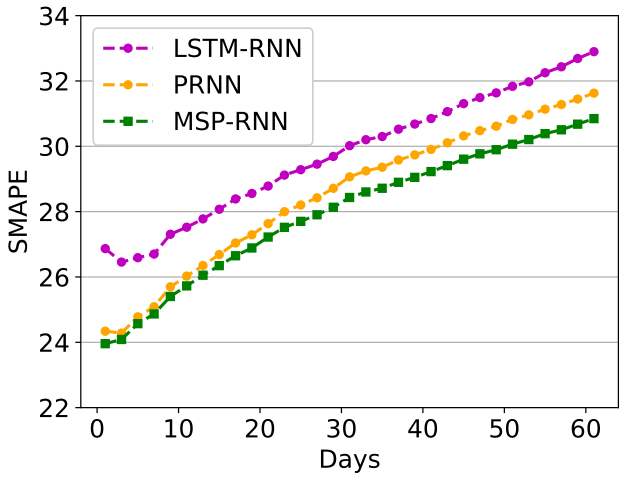


Fig. 9: SMAPE of LSTM-RNN, PRNN and MSP-RNN with growing prediction days.

5.4.2 *Impact of Cluster Number*

In MSP-RNN, the piecewise regressive mechanism is the most important component to enhance the prediction accuracy, so we meticulously study the influence of cluster number predefined. We conducted 10 repeated experiments with different cluster numbers varying from 1 to 10, and the SMAPE of these experiments are drawn in Fig. 10. In this line chart, SMAPE declines with the increasing cluster number predefined in general. When $n_{\text{class}} = 2$ or 3, the variance of SMAPE is large, which indicates these hyper-parameters are inappropriate for MSP-RNN, because the regularization items may make it difficult to converge. When $n_{\text{class}} \leq 4$, SMAPE fluctuates, which demonstrates that increasing predefined cluster number cannot bring about further improvement of prediction accuracy.

To determine the optimal cluster number predefined, we carefully studied the intermediate results of MSP-RNN. Taking one experiment as an example, Fig. 11 shows the distribution of sequences that were classified into different categories. In this experiment, we predefined cluster number as 5. As shown in this figure, MSP-RNN mainly divides them into 4 clusters, i.e., class #0, #1, #3, #4, which

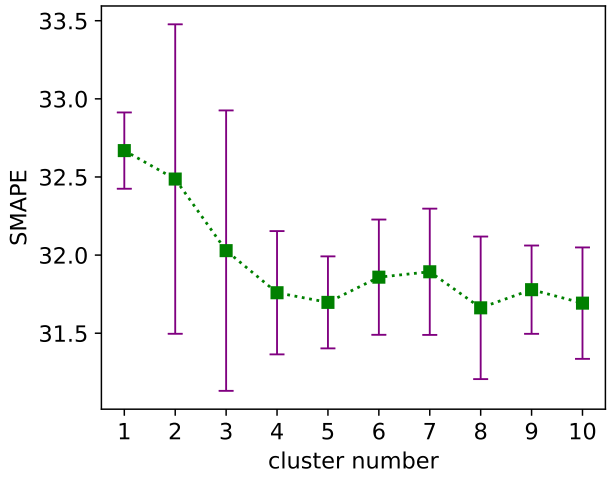


Fig. 10: Error bar of SMAPE with growing cluster numbers.

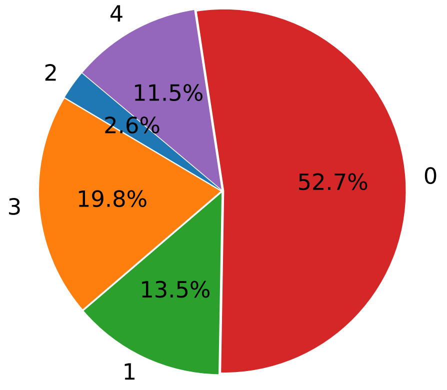


Fig. 11: Class distribution of prediction results when cluster number was predefined as 5.

account approximately 90% of the results. Actually, this is not an accident. Through extensive experiments, we found that even the cluster number predefined is 9, MSP-RNN can only converge to 4 main categories. We think this might due to the unbalanced data distribution, or limited cluster way.

To further study the piecewise regressive mechanism, we visualized the weight of various fully connected layers in Fig. 13 to study how MSP-RNN treats different types of sequences. Fig. 13 (a) shows the weight parameter of fully connected layers between the autocorrelation function and the softmax function, where we can find different clusters paid different attention to the autocorrelation coefficients. More specifically, all clusters focus largely on the initial autocorrelation coefficients, which is very reasonable because most sequences are affected greatly by their past few days, while different clusters also paid different attentions to different autocorrelation coefficients. As for the decoding manner, we observed that some hidden states (e.g., #31, #36 and #100) are highly valued by several decoders, indicating that these hidden states are very useful. We also observed that different decoders utilize the hidden states discriminatingly due to the piecewise regressive mechanism, which breeds the diversity. In Section 5.4.4, we will produce several cases to demonstrate the clustering results and point out the advantages and disadvantages that our model presents in different situations.

Through the above analysis, we deem that $n_{\text{class}} = 5$ is the best for MSP-RNN, because it is just adequate for the softmax classifier to separate meaningful clusters without wasting much computational resources. However, we also believe that if better hand-craft features or cluster way are applied, more meaningful patterns could be recognized , rendering this hyper-parameter larger, and eventually the effect of the piecewise regressive mechanism will become stronger.

5.4.3 Impact of Step Length

Except for the piecewise regressive mechanism, the multi-step prediction strategy is also an essential part to enhance the quality of prediction of our MSP-RNN model. In the experiments, we tested MSP-RNN with the prediction step length from 1 to 5 steps, and reported the results in Fig. 12. As shown in this line chart, SMAPE decreases with the step length predefined increasing, while it rebounds when $n_{\text{step}} = 4$. This indicates that the multi-step prediction strategy can help enhance prediction accuracy, if a slight n_{step} is set. We deem such improvement is due to the principle of LSTM (Section 3.1) that RNN is likely to attach greater importance to current input x_t rather than previous states $\boldsymbol{h}_{t-1}$, while the multi-step training strategy may eliminate such dependencies in some way, and allow RNN to value more on the previous states. However, if a large n_{step} is set, MSP-RNN might also be confused, and lead to a worse result.

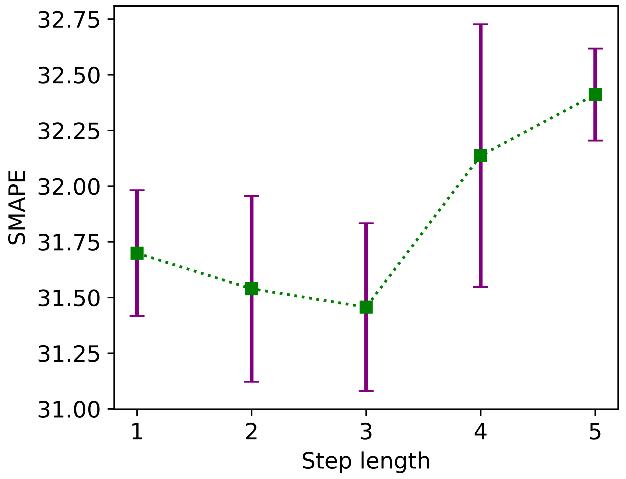
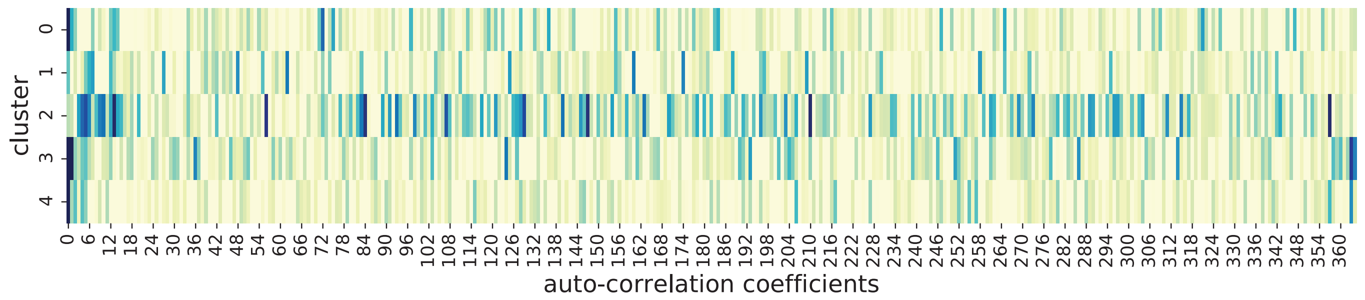


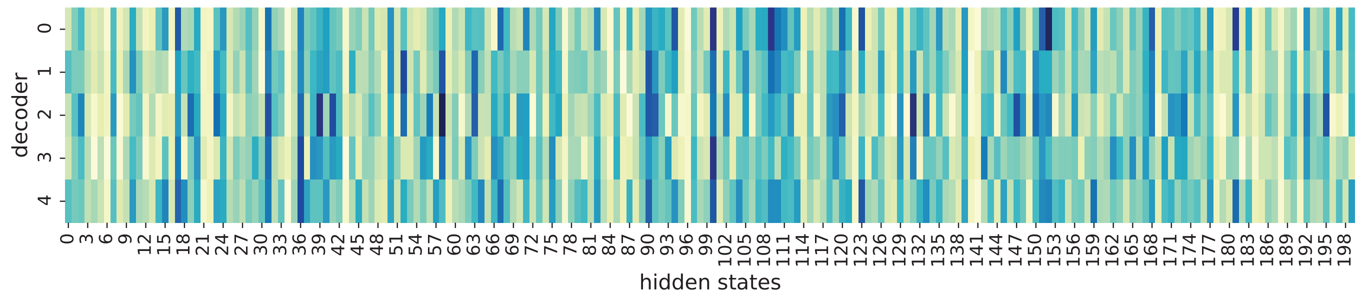
Fig. 12: SMAPE of MSP-RNN with different steps.

5.4.4 Case Study

To further study the merits and demerits of MSP-RNN and other baseline methods, we have observed substantive



(a) Attention to different autocorrelation coefficients for different clusters.



(b) Attention to different hidden states from LSTM for different fully connected layers.

Fig. 13: Visualization of fully connected layers weights in MSP-RNN.

page views, their prediction results from different models and their corresponding autocorrelation coefficients. In this section, we will analyze some of them, which are shown in Fig. 14.

Case 1. Fig. 14 (a) shows the page views of wikepage *Romania*[7] in the upper part and its corresponding autocorrelation coefficients in the lower part, which is also a typical example of $\#0$ cluster with a 87% probability determined by MSP-RNN. For this cluster, the autocorrelation coefficients concentrate on 0, indicating that it shows minute correlation with its past records. Therefore, we deem that this cluster is like white noise. Through our experiments, MSP-RNN as well as other baseline methods all performed quite well in this situation. In this case, the SMAPE of ARIMA, SVR, LSTM-RNN, LSTNet and MSP-RNN are 10.96, 7.93, 6.54, 6.67, 6.34, respectively.

Case 2. Fig. 14 (b) is about NCIS[8], a TV series, and it is a representative of #1 cluster with 91% likelihood. Just like the heatmap of the fully connected layer of cluster #1 (see Fig. 13 (a)), this cluster values autocorrelation in many places, where the peaks show. In this case, the prediction of ARIMA goes an entirely wrong direction, which is because this model is quite easily affected by its past few days, while the LSTM-based models predict right tendency. For this case, SMAPE of ARIMA, SVR, LSTM-RNN, LSTNet and MSP-RNN are 27.66, 10.83, 8.36, 9.24, 8.17, respectively.

Case 3. Fig. 14 (c) is concerning Caesium carbonate[9], a kind of chemical substance that is a representative of cluster #2. Similar with the heatmap of the fully connected layer of cluster #2, this cluster also pays attention to autocorrelation in many places. Although cluster #2 only accounts for 2.6% in all sequences, MSP-RNN frequently outperforms other baseline methods in this situation.

Case 4. Fig. 14 (d) introduces 76mm air defense gun M1938[10], which is a representative of cluster #3. Fig. 14 (e) introduces Facebook[11], which is another representative of cluster #4. Their autocorrelation coefficients do associate with the weight of corresponding fully connected layers, but they seem not so meaningful. Even so, our empirical results show that MSP-RNN outperforms other methods frequently in this situation. But, we also believe that a better cluster way may be required here to produce more interpretable clustering results.

Case 5. All above cases show the most typical examples of 5 kinds of clustering results, and in many situations, MSP-RNN does perform well. However, MSP-RNN is far more than that. Fig. 14 (f) is about YouTube[12], and its probabilities to different categories calculated by MSP-RNN are 43%, 2%, 2%, 49% and 2%, respectively. For this case, both decoder #0 and decoder #3 will mainly contribute to the results. In this case, SMAPE of ARIMA, SVR, LSTM-RNN, LSTNet and MSP-RNN are 12.25, 15.54, 10.24, 7.75, 6.90, respectively. Actually, most of the experimental sequences will be predicted by more than one decoder (fully connected layer), and these decoders also make the results more accurate.

As illustrated in the above analyses, we have proved that MSP-RNN outperforms other baseline methods in term of

7. https://en.wikipedia.org/wiki/Romania
8. https://en.wikipedia.org/wiki/NCIS_(TV_series)
9. https://en.wikipedia.org/wiki/Caesium_carbonate
10. https://en.wikipedia.org/wiki/76_mm_air_defense_gun_M1938
11. https://en.wikipedia.org/wiki/Facebook
12. https://en.wikipedia.org/wiki/YouTube

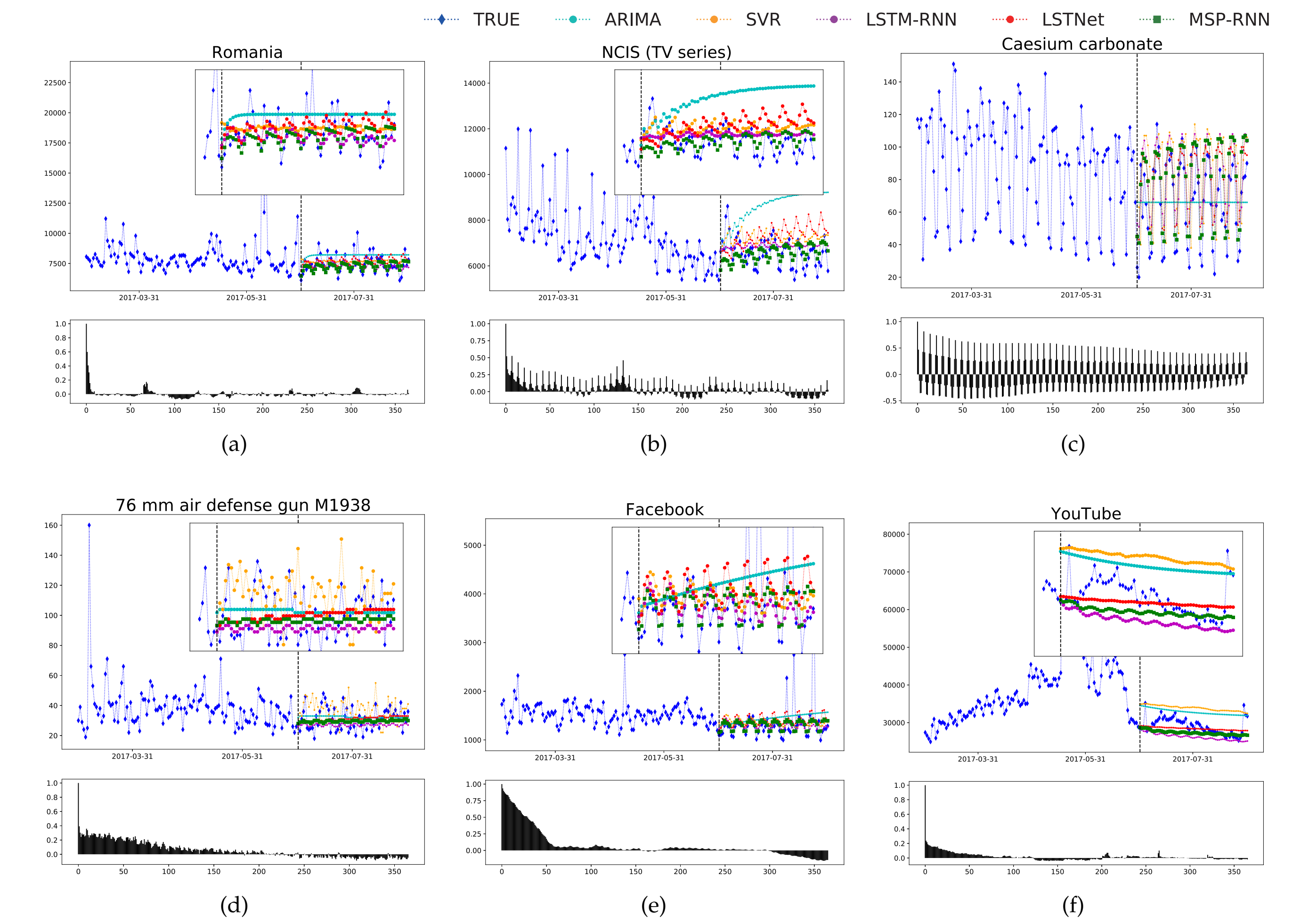


Fig. 14: Typical cases. The upper part of each case shows true values and predicted ones, where blue line represents the true page views of a wikipage, and cambridge blue, orange, purple and green lines represent the prediction results of ARIMA, SVR, LSTM-RNN and MSP-RNN, respectively. It is noticeable that the blue lines before the dashed line (July 1, 2017) in the upper parts represent real data for training models during a year, while the rest after the dash line are used to evaluate models. The lower part shows autocorrelation coefficients corresponding to the upper cases, which is calculated by using only training data.

prediction accuracy. Meanwhile, its potential still needs to be further developed.

6 Related Work

As our work touches on a couple of different research areas, in this section, we will review some representative ones and differentiate our work from them.

6.1 QoS Inference

When it comes to predicting the tendency of service invocation, one may associate with predicting Quality of Services (QoS) that includes a set of indicators, like packet loss, bit rate, throughput, transmission delay, etc. Although there exist some similarities between these indicators and tendency of services invocation, actually they are quite different.

With respect to methodology, existing works concerning QoS predicting aim to infer a score about how good a user can receive a service from its provider. Initially, many works have not considered the factor of changing time. In this scenario, non-negative matrix factorization has been widely applied. Based on this, Y. Zhang et al. [27] and Y. Ma el al. [28] have built the relationship between user-to-user and service-to-service, and successfully inferred the missing QoS values. Further, W. Zhang et al. [29] have considered the time factor and based on tensor factorization to infer the missing QoS values with changing time. However, the changes of these QoS indicators in future days still cannot be predicted. Subsequently, some studies have considered the factor of time and make prediction. Based on Autoregressive Moving Average (ARMA) model, M. Godse et al. [30] and Amin et al. [31] have proposed their own methods to predict individual QoS for one user. To build the relationship between users and services, R. Xiong et al. [32] have built an LSTM-based model to predict QoS dynamically.

Besides, all above methods are facing a fixed user base, while our problem predicting the tendency of services invocation is confronting a service base. That is to say, to solve the problem, we can only utilize the past invocation records without users relationship. Additionally, the number of services being huge and the traits of them being vastly different also make our problem more challenging.

6.2 Time Series Prediction

Time series prediction has always been a critical research problem, and it goes through a long development history. Traditionally, people tended to analyze the traits of time series and designed corresponding models to predict ones separately. In 1970, GEP Box et al. [7] have first put forward ARMA and Autoregressive Integrated Moving Average (ARIMA) model for predicting time series being stationary and non-stationary. These models can effectively model the linearity and periodicity of time series, and still take a dominant position in this domain. However, since many time series are nonlinear, later works are proposed to cover this point. Z. Ding [8] and TB Trafalis et al. [33] have applied Support Vector Machine to Regress (SVR) stock price, which makes up for the limitation of linear models. These models can be solutions to predict the tendency of service invocation in our problem.

However, these models are univariate models. When facing multivariate prediction problem, the correlation between variables will be neglected, meaning that they cannot utilize the intrinsic relationship among services. Subsequently, Vector Autoregression (VAR) model and Multiple-output SVR, extension of ARMA and SVR model, have been proposed to seize the correlation between sequences [34], [35]. Since this model takes immense computer memory to build relationships, it cannot address time series prediction problem with thousands ones. As a consequence, our problem cannot be solved by them.

In recent years, big data has been becoming a popular topic, which results in a need to predict time series in a very large scale data. In this context, a substantial amount of data on past days can be leveraged for predicting thousands time series. Lee et al. [36] have used RBF neural network to predict time series. Dasgupta et al. [37] have proposed Nonlinear Dynamic Boltzmann Machines to predict time series. Specially, LSTM and GRU are capable of remembering long short term dependencies among a sequences thus have been widely used [12], [38]. Recently, attention mechanism has also been applied to capture the temporal dependency of time series, and gain great strides in prediction accuracy [25], [39]. However, because our problem presents a combination dependencies *(i.e., mashups invoking their comprising services)*, these models still cannot fully solve our problem.

6.3 Time Series Clustering

Time series clustering is another important research area related to our work. Studying how to cluster time series can facilitate predicting millions of time series. In this section, we review several famous works about time series clustering. To begin with, from Aghabozorgi's point of view, approaches to clustering time series can be divided into three categories: whole time-series clustering, sub-sequence clustering and time point clustering [14]. Among them, the whole time-series clustering refers to dividing time series into categories with respect to their similarity, which can be further divided into model-based approaches, feature-based approaches and shape-based approaches.

Model-based approaches refer to people predefining a parametric model, then transforming raw time series date into model parameters, and finally clustering them according to model distance. For example, Bicego et al. [40] and Panuccio et al. [41] have applied Hidden Markov Model (HMM) to cluster time series. Feature-based approaches refer to transforming raw time series date into some particular features and then applying clustering algorithm on these features. Piecewise regressive mechanism is one of them, which transforms raw time series into autocorrelation coefficients, and then learns to cluster them through a neural network.

Besides, shape-based approaches refer to making assignment according to similarity between two time series. Dynamic Time Warping (DTW), which we adopt to evaluate similarity of two time series, is a typical method to align time series. Similarly, Niennattrakul et al. [42] have clustered multimedia time series by using k-means algorithm and DTW. Some other works have used other distances to cluster time series. For example, Shumway et al. [43] have introduced J divergence; Chen et al. [44], [45] have introduced Edit Distance; Zhang el al. [46] have introduced Triangle similarity. Inspired by these clustering methods, we merge an effective one into our model, especially the loss function, thus to predict the tendency of services invocation more accurately with respect to shape in particular.

7 Conclusions

As predicting the tendency of services invocation has been progressively beneficial to all roles in service ecosystems, in this paper, we abstract three unique features of services invocation, namely complicated traits, intricate relationship and combination dependencies. Tailored by these features, we have developed a deep neural network, named Multi-Step Piecewise Recurrent Neural Network (MSP-RNN), to predict the tendency of services invocation, whose prediction accuracy is higher than our previous work Piecewise Recurrent Neural Networks (PRNN) and other state-of-the-art baseline methods. The main ideas include: 1) applying LSTM units to model the complicated characteristics of service invocation sequences generally; 2) developing a piecewise mechanism to predict different types of service invocation sequences discriminatingly; and 3) proposing a multi-step prediction strategy to enhance model robustness and further increase prediction accuracy. Extensive experiments with interpretable analysis have proved that MSP-RNN significantly outperforms the baseline methods in term of prediction accuracy, i.e., by 3.7% with respect to SMAPE, particularly, when the prediction days become longer.

In our future research, we plan to focus on the following three aspects: 1) to lucubrate the piecewise regressive mechanism to better cluster and predict service invocation sequences; 2) to model the combination dependencies and further increase the accuracy of predicting the tendency of services invocation; and 3) to study online predicting technology, so that the tendency of services invocation can be predicted in a shorter time period.

ACKNOWLEDGMENTS

This research has been partially supported by the National Key Research and Development Program of China (No. 2018YFB1402500), the National Nature Science Foundation of China (No.61673230) and High-Tech Ship Research Project of China (17GC26102.01). Yushun Fan is the corresponding author.

REFERENCES

[1] Y. Wei and M. B. Blake, "Service-oriented computing and cloud computing: Challenges and opportunities," *IEEE Internet Computing*, vol. 14, no. 6, pp. 72–75, 2010.

[2] J. Zhang, W. Tan, J. Alexander, I. Foster, and R. Madduri, "Recommend-as-you-go: A novel approach supporting services-oriented scientific workflow reuse," in *IEEE International Conference on Services Computing*, 2011, pp. 48–55.

[3] A. E. Mrquez-Chamorro, M. Resinas, and A. Ruiz-Corts, "Predictive monitoring of business processes: a survey," *IEEE Transactions on Services Computing*, 2017.

[4] K. Hashmi, Z. Malik, A. Erradi, and A. Rezgui, "Qos dependency modeling for composite systems," *IEEE Transactions on Services Computing*, 2016.

[5] K. Li, J. Mei, and K. Li, "A fund-constrained investment scheme for profit maximization in cloud computing," *IEEE Transactions on Services Computing*, 2016.

[6] B. Bai, Y. Fan, W. Tan, and J. Zhang, "Sr-lda: Mining effective representations for generating service ecosystem knowledge maps," in *IEEE International Conference on Services Computing*, 2017, pp. 124–131.

[7] G. E. P. Box and G. M. Jenkins, *Time series analysis forecasting and control*. HOlden-Day, 1970.

[8] Z. Ding, *Application of Support Vector Machine Regression in Stock Price Forecasting*. Springer Berlin Heidelberg, 2012.

[9] R. J. Williams, G. E. Hinton, and D. E. Rumelhart, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, p. 17351780, 1997.

[11] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.

[12] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to time series predictable through time-window approaches," in *International Conference on Artificial Neural Networks*, 2001, pp. 669–676.

[13] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[14] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, *Time-series clustering - A decade review*. Elsevier Science Ltd., 2015.

[15] H. Lin, Y. Fan, J. Zhang, and B. Bai, "PRNN: Piecewise recurrent neural networks for predicting the tendency of services invocation," in *IEEE International Conference on Web Services*, 2018, pp. 42–49.

[16] B. Bai, Y. Fan, W. Tan, and J. Zhang, "Dltsr: A deep learning framework for recommendation of long-tail web services," *IEEE Transactions on Services Computing*, no. 1, pp. 1–1, 2017.

[17] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," *Journal of the Operational Research Society*, vol. 66, no. 8, pp. 1352–1362, 2015.

[18] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[19] J. Mackinnon, "Approximate asymptotic distribution functions for unit-root and cointegration tests," *Working Papers*, vol. 12, no. 2, pp. 167–176, 1992.

[20] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE transactions on intelligent transportation systems*, vol. 5, no. 4, pp. 276–281, 2004.

[21] K.-j. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, 2003.

[22] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

[23] M. A. Zaytar and C. El Amrani, "Sequence to sequence weather forecasting with long short term memory recurrent neural networks," *Int J Comput Appl*, vol. 143, no. 11, 2016.

[24] K. Chen, Y. Zhou, and F. Dai, "A lstm-based method for stock returns prediction: A case study of china stock market," in *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2823–2824.

[25] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 2018, pp. 95–104.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based qos prediction in cloud computing," in *IEEE International Symposium on Reliable Distributed Systems*, 2011, pp. 1–10.

[28] Y. Ma, S. Wang, F. Yang, and R. N. Chang, "Predicting qos values via multi-dimensional qos data for web service recommendations," in *IEEE International Conference on Web Services*, 2015, pp. 249–256.

[29] W. Zhang, H. Sun, X. Liu, and X. Guo, "Temporal qos-aware web service recommendation via non-negative tensor factorization," in *International Conference on World Wide Web*, 2014, pp. 585–596.

[30] M. Godse, U. Bellur, and R. Sonar, "Automating qos based service selection," in *IEEE International Conference on Web Services*, 2010, pp. 534–541.

[31] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting qos attributes of web services based on arima and garch models," in *IEEE International Conference on Web Services*, 2012, pp. 74–81.

[32] R. Xiong, J. Wang, Z. Li, B. Li, and P. C. K. Hung, "Personalized lstm based matrix factorization for online qos prediction," in *IEEE International Conference on Web Services*, 2018, pp. 34–41.

[33] T. B. Trafalis and H. Ince, "Support vector machine for regression and applications to financial forecasting," in *Ieee-Inns-Enns International Joint Conference on Neural Networks*, 2000, p. 6348.

[34] H. Ltkepohl, *New Introduction to Multiple Time Series Analysis*. Springer, 2005.

[35] Y. Bao, T. Xiong, and Z. Hu, *Multi-step-ahead time series prediction using multiple-output support vector regression*. Elsevier Science Publishers B. V., 2014.

[36] C. M. Lee and C. N. Ko, "Time series prediction using rbf neural networks with a nonlinear time-varying evolution pso algorithm." *Neurocomputing*, vol. 73, no. 1, pp. 449–460, 2009.

[37] S. Dasgupta and T. Osogami, "Nonlinear dynamic boltzmann machines for time-series prediction," in *AAAI Conference on Artificial Intelligence*, 2016.

[38] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, no. 1, 2018.

[39] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 2, pp. 601–614, 2018.

[40] M. Bicego, "Similarity-based clustering of sequences using hidden markov models," in *International Conference on Machine Learning and Data Mining in Pattern Recognition*, 2003, pp. 86–95.

[41] A. Panuccio, M. Bicego, and V. Murino, "A hidden markov model-based approach to sequential data clustering," vol. 2396, pp. 734–742, 2002.

[42] V. Niennattrakul and C. A. Ratanamahatana, "On clustering multimedia time series data using k-means and dynamic time warping," in *International Conference on Multimedia and Ubiquitous Engineering*, 2007, pp. 733–738.

[43] R. H. Shumway, "Time-frequency clustering and discriminant analysis," *Statistics & Probability Letters*, vol. 63, no. 3, pp. 307–314, 2003.

[44] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," 2004, pp. 792–803.

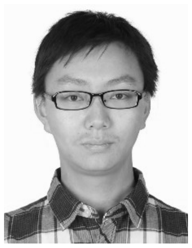
[45] L. Chen and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. 2005 ACM SIGMOD International Conference on Management of Data*, 2005, pp. 491–502.

[46] X. Zhang, J. Wu, X. Yang, H. Ou, and T. Lv, "A novel pattern extraction method for time series classification," *Optimization & Engineering*, vol. 10, no. 2, pp. 253–271, 2009.

Haozhe Lin received the B.S degree in from Central South University, China, He is currently working toward the Ph.D degree at the Department of Automation, Tsinghua University. He received the Best Student Paper Award from the 2018 IEEE International Conference on Web Services. His research interests include services computing, time series prediction, and data mining.

Yushun Fan received the Ph.D. degree in control theory and application from Tsinghua University, China, in 1990. He is currently a tenure professor with the Department of Automation, Vice Director of National CIMS Engineering Research Center of China, Director of the System Integration Institute, and Director of the Networking Manufacturing Laboratory, Tsinghua University. He is the member of IFAC TC5.1 and TC 5.2, Vice director of China Standardization Committee for Automation System and Integration. Editorial member of the International Journal of Computer Integrated Manufacturing, From September 1993 to 1995, he was a visiting scientist, supported by Alexander von Humboldt Stiftung, with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany. He has authored 10 books in enterprise modeling, work-flow technology, intelligent agent, object-oriented complex system analysis, and computer integrated manufacturing. He has published more than 500 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process re-engineering, workflow management, system integration, modern service science and technology, petri nets modeling and analysis.

Jia Zhang received the M.S. and B.S. degrees in computer science from Nanjing University, China and the PhD degree in computer science from the University of Illinois at Chicago. She is currently an associate professor at the Department of Electrical and Computer Engineering, Carnegie Mellon University. Her recent research interests center on service oriented computing, with a focus on collaborative scientific workflows, Internet of Things, cloud computing, and big data management. She has co-authored one textbook titled “Services Computing“ and has published more than 130 refereed journal papers, book chapters, and conference papers. She is currently an associate editor of the IEEE Transactions on Services Computing (TSC) and of International Journal of Web Services Research (JWSR), and editor-in-chief of International Journal of Services Computing (IJSC). She is a senior member of the IEEE.



Bing Bai received the B.S. and Ph.D. degree in control theory and application from Tsinghua University, China, in 2013 and 2018 respectively, and he is currently a senior researcher with the Cloud and Smart Industries Group, Tencent, Beijing, China. He received the Best Paper Award from the 14th IEEE International Conference on Services Computing (2017). His research interests include data mining and recommender systems.