PRNN: Piecewise Recurrent Neural Networks for Predicting the Tendency of Services Invocation

Haozhe Lin
Tsinghua National Laboratory for Information Science and Technology Department of Automation Tsinghua University Beijing 100084, China
linhz16@mails.tsinghua.edu.cn

Yushun Fan[*]
Tsinghua National Laboratory for Information Science and Technology Department of Automation Tsinghua University Beijing 100084, China
fanyus@tsinghua.edu.cn

Jia Zhang
Department of Electrical and Computer Engineering Carnegie Mellon University Silicon Valley Moffett Field CA 94035, USA
jia.zhang@sv.cmu.edu

Bing Bai
Tsinghua National Laboratory for Information Science and Technology Department of Automation Tsinghua University Beijing 100084, China
bb13@mails.tsinghua.edu.cn

***Abstract*—Driven by the widespread application of Service-Oriented Architecture (SOA), the quantity of web services and their users keeps increasing in the service ecosystem. Since services are hosted by service providers, it will be very helpful to predict the tendency of services invocation for service providers, so that proper actions may be taken to ensure the quality of services. Two major challenges exist in predicting the tendency of services invocation, however. First, different service invocation sequences may bear different and complicated characteristics, which is hard to be modeled generally. Second, the intricate relations between service invocation sequences are valuable but hard to be discriminated and utilized. To address these issues, a deep neural network, named Piecewise Recurrent Neural Network (PRNN), is developed by taking both generality and pertinence into consideration. For generality, PRNN extracts complicated characteristics of all service invocation sequences through Long Short-Term Memory (LSTM) units. For pertinence, PRNN develops a piecewise mechanism, through which service invocation sequences can be clustered automatically and predicted discriminatingly. Extensive experiments in real-world dataset show that PRNN outperforms baseline methods in predicting the tendency of services invocation.**

***Keywords*-Service discovery; tendency of services invocation; LSTM; piecewise mechanism**

I. INTRODUCTION

With the rapid adoption of Service-Oriented Architecture, numerous services have been published onto the Internet and people have been leveraging available services to create value-added new services, making the service ecosystem prosperous [1]. However, it gives rise to the difficulties for service providers in maintaining a stable Quality of Service (QoS). Since service providers are not facing fixed user base, service usage patterns may fluctuate over time, and thus a static strategy may not work. For example, some providers may rent too many servers to avoid possible network congestions and bear a high cost, while others may rent too few and suffer from losing users for their delayed service supplement at peak time. Obviously, understanding and predicting the **tendency of services invocation** *(i.e., the number of services invocation at a given time period)* is becoming increasingly important for service providers to adjust their operating strategy dynamically and achieve a desirable QoS in an economic way.

[*] Corresponding Author

To predict the tendency of services invocation, some non-negligible facts require special attention. 1) Different service invocation sequences present different and complicated characteristics, like nonlinearity, periodicity and long-term dependency. Take *Sportradar Olympics API* as an example. Because of its Olympics relevance, the API always reaches a peak time when Olympics is held, and thus shows a four-year periodicity during the Games without distinct patterns in other days. Such a usage pattern is obviously unique from many other services. 2) Services with similar functions tend to present similar invocation tendency, and such similarities have potential for predicting the tendency of services invocation. For example, *Google Maps* is one of the most famous mapping function services, while *Mappy* is an analogous but obscure one, consequently the invocation tendency of them being likely to be similar [2]. If this relation is utilized, the changes of *Google Maps* will be helpful for predicting the invocation tendency of *Mappy* and vice versa. 3) Because of the particular existence of mashups (e.g., services combination) in the service ecosystem, some services are always invoked at the same time when their mashup is invoked, which makes the relations of service invocation sequences more complicated. For instance, *Foodsta* is a mashup comprised of two APIs: *Instagram* and *Google Maps*. Apart from being invoked alone, *Instagram* and *Google Maps* are invoked at the same time when *Foodsta* is invoked, which increases the difficulty in predicting the tendency of services invocation. These intractable facts remarkably differ the service invocation prediction problem from other time-series prediction problems, and thus further study is demanded in predicting the tendency of services

978-1-5386-7247-1/18/$31.00 ©2018 IEEE
DOI 10.1109/ICWS.2018.00013

invocation.

Recently, Recurrent neural network (RNN) has been arguably regarded as one of the best models for sequences modeling, for its capacity of capturing the general changing patterns by learning from enormous data [3], [4]. Since service invocation records can be represented as thousands of sequences, RNN can be used to predict the tendency of services invocation in an acceptable accuracy. In spite of its presenting a cogent generalization, however, RNN treating all sequences as a whole impairs the pertinence for prediction and exposes some limitation. On the contrary, some classic univariate models focusing on modeling time series individually, like ARMA [5], can diminish such limitation. However, the valuable relations between services will be wasted, if those models are applied. Intuitively, taking both generality and pertinence into account may enhance service invocation prediction. In such a context, we have developed a novel deep neural network named Piecewise Recurrent Neural Networks (PRNN) to predict the tendency of services invocation.

On the one hand, PRNN aims to capture complicated characteristics, such as nonlinearity, periodicity and long-term dependency, of service invocation sequences in general. Be more specific, PRNN applies Long Short-Term Memory (LSTM), a special structure of RNN, to treat all the service invocation sequences as the same to encode them into hidden states. After sufficient training, such hidden states will capture the changing patterns of different kinds of service invocation sequences and will serve as effective features for service invocation prediction. To further enhance the prediction accuracy, on the other hand, PRNN develops a piecewise mechanism by taking into consideration the pertinence of service invocation sequences. Concretely, PRNN trains several parallel fully connected layers to decode the hidden states into intermediate results. At the same time, PRNN takes autocorrelation coefficients of service invocation sequences as input of a softmax function to classify them into different clusters. During the training process, PRNN adjusts the methods of clustering and decoding automatically until it finds the best equilibrium. As a result, by combining these intermediate results, homogeneous service invocation sequences will be predicted discriminatingly.

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

1) We apply LSTM units to extract complicated characteristics of service invocation sequences. After sufficient training, difficult features can be extracted by PRNN effectively and generally.
2) We propose a piecewise mechanism to take the pertinence into consideration. Taking the autocorrelation coefficients as priority, several trained fully connected layers can decode the hidden states diversely and PRNN can predict the tendency of services invocation more discriminatingly.
3) Extensive experiments over real-world dataset show that PRNN outperforms baseline methods in terms of prediction accuracy. Particularly, PRNN performs far better than baseline methods when predicting a longer tendency of services invocation.

The remainder of this paper is organized as follows. Section II describes the framework of our PRNN model. Section III introduces the details of parameter learning. Section IV reports the experimental results. Section V discusses the related work. Finally, Section VI draws conclusions.

II. Model Framework

In this section, we firstly restate the problem mathematically, and then propose a neural and probabilistic framework to predict the tendency of services invocation in the future.

A. Notation and Problem Definition

Throughout the rest of this paper, we use $\mathbf{x}_i$ to denote a service invocation sequence for service i in the service ecosystem, which can be broken down to $\mathbf{x}_i = \{x_{i0}, x_{i1}, \dots, x_{in}\}$ for service i in the past n days, where $x_{ik}(k \in n)$ represents the number of invocations on service $\mathbf{x}_i$ on day k. Likewise, we use $\hat{\mathbf{y}}_i$ to denote a prediction sequence for service i, and use $\hat{\mathbf{y}}_i = \{\hat{y}_{i1}, \hat{y}_{i2}, \dots, \hat{y}_{i(t+1)}\}$ to represent the predicted number of invocations for service i in the next t days.

Assuming that there are m services in the service ecosystem, we define the problem as using m service invocation sequences within n days to predict m sequences as tendency of services invocation in the next t days, reaching a prediction accuracy as high as possible.

To solve this problem, we propose a deep neural network, named Piecewise Recurrent Neural Networks (PRNN), taking $\mathbf{x}_i$ as input and generating $\hat{\mathbf{y}}_i$ as the prediction, whose overall framework is depicted in Fig. 1.

B. LSTM-based Feature Extraction

The prerequisite of predicting the tendency of services invocation is to capture the past changing patterns. Since the characteristics of service invocation sequences are extremely complicated and the number of them is enormous, it is natural to introduce the Long Short-Term Memory (LSTM), a special structure of RNN, to extract features generally [6]. As shown in Fig. 2, an LSTM unit consists of a cell c acting as a memory and three gates (input, output, forget) enabling modification of the cell memory, which can be described as Eq. 1 – 6 mathematically.

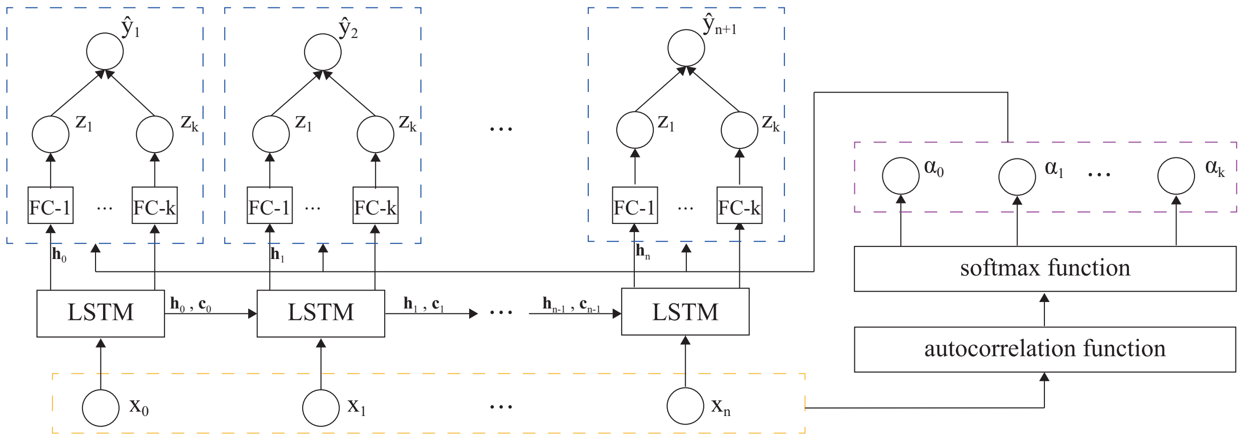


Figure 1. An overall framework of PRNN. In the orange dashed rectangle, a batch of service invocation sequences $\{x_0, x_1, \ldots, x_n\}$ are used as input of both LSTM units and autocorrelation function. On the one side, PRNN applies LSTM units to encode them into hidden states $\mathbf{h}$ in every time step. On the other side, in the purple dashed rectangle, PRNN calculates the corresponding probabilities α by utilizing an autocorrelation function and a softmax function. In the blue dashed rectangle, PRNN first trains several parallel fully connected layers (FC) to decode the hidden states into intermediate results z and then combine them based on the probabilities to generate the final prediction in every time step.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{1}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$
$$\widetilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{3}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t \tag{4}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

where $\odot$ represents the dotwise product, W_i, W_f, W_o and W_c matrices are parameters of gates and the memory cell, $\sigma(\cdot)$ and $\tanh(\cdot)$ are sigmoid function and hyperbolic function, respectively.

Once receiving sufficient training, LSTM will extract the complicated characteristics of service invocation sequences. Based on these effective features (e.g., the hidden states from LSTM), a fully connected layer is able to decode them into prediction values and present an acceptable prediction accuracy theoretically [7].

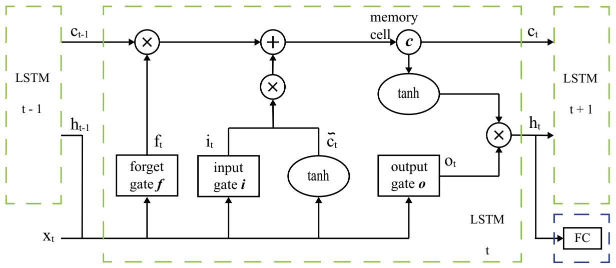


Figure 2. An LSTM unit in the context of PRNN. Memory cell c records historical information, and three gates $(\mathbf{f}, \mathbf{i}, \mathbf{o})$ execute sigmoid functions to modify the memory at every time step. The hidden states will be used as input of both LSTM units and fully connected layers (FC).

C. Autocorrelation-based Piecewise Prediction

To further improve the prediction accuracy, we make a reasonable assumption that different kinds of service invocation sequences have different expression ways from hidden states, so training several parallel fully connected layers may be helpful to decode such features in different ways. Thus, we propose a piecewise mechanism to predict the tendency of services invocation discriminatingly, which idea is illustrated in Fig. 3.

On the one hand, to express the features in different ways, PRNN trains several parallel fully connected layers (e.g., FC in Fig. 3) contraposing different types of sequences to decode hidden states from LSTM into intermediate results, which can be described in Eq. 7.

$$z_k = \mathbf{w}_k^\top \cdot \mathbf{h} \tag{7}$$

where $\mathbf{h}$ is the hidden states extracted by LSTM units, various $\mathbf{w}$ are weight parameters of different fully connected layers and z_k is the intermediate result.

On the other hand, to avoid the trained fully connected layers being too similar, PRNN applies an autocorrelation function on service invocation sequences to obtain the auto-correlation coefficients and distinguish sequences effectively without being affected by the order of magnitude or extreme values compared with other statistics in Eq. 8. Then based on these coefficients, PRNN uses a softmax function to classify service invocation sequences into different clusters and calculate the corresponding probabilities in Eq. 9.

$$\rho(\tau) = \frac{\mathbb{E}\left[(x_t - \mu)(x_{t+\tau} - \mu)\right]}{\sigma^2} \tag{8}$$

$$\alpha_i = \frac{e^{\boldsymbol{\omega}_i^\top \cdot \boldsymbol{\rho}}}{\sum_{k=1}^{n_\text{class}} e^{\boldsymbol{\omega}_k^\top \cdot \boldsymbol{\rho}}} \tag{9}$$

where $\rho(\tau)$ is the autocorrelation coefficient with time lag τ, $\mathbb{E}$ is the expected value operator, μ is the mean value of a service invocation sequence, σ^2 is the variance of a service invocation sequence, x_t is the invocation records of a service in day t, various $\boldsymbol{\omega}$ are the weight parameters affecting classification, $\boldsymbol{\alpha}$ is probability of service invocation sequence belonging to different clusters, and the class number n_{class} is the number of the clusters predefined.

Through the piecewise mechanism, PRNN will find the equilibrium between the cluster way and fully connected layers parameters ceaselessly during the training process, and will achieve the best prediction accuracy automatically. For example, service invocation sequences with weekly periodicity may be clustered into a category, while sequences with other characteristics may be clustered into another. In other words, some tailored fully connected layers will decode the hidden states as features into intermediate prediction result discriminatingly. As a result, service invocation sequences can be predicted discriminatingly in Eq. 10 and the prediction accuracy will be improved.

$$\hat{y} = \sum_{k=1}^{n_{\text{class}}} \alpha_k \cdot z_k \tag{10}$$

where α_k is the probability of a sequence belonging to class k that can be calculated by Eq. 9, n_{class} is the number of fully connected layers and $\hat{y}$ is the prediction value generated by PRNN.

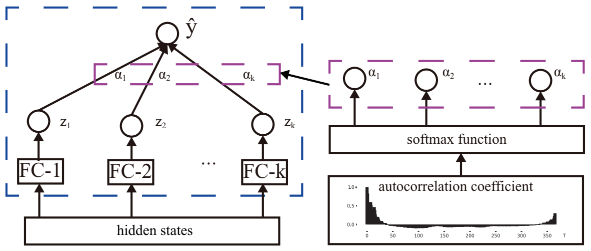


Figure 3. A diagram of piecewise regression. Various z_k are intermediate results from different fully connected layers, and the combination of them will generate the prediction value. We apply autocorrelation coefficients as input of a softmax function, and take the output probability as the weight for the combination.

III. Parameter Learning

In this section, we design a tailored loss function and introduce some delicate setting to ensure PRNN work.

A. Loss Function

To achieve the best prediction accuracy, we design a loss function (Eq. 11) for PRNN from two perspectives.

Because of the great quantity of services and differences in the order of magnitude between them, many common indicators, like mean square error (MSE) and mean absolute error (MAE), cannot reflect the prediction accuracy properly. It is apparent that a service invocation sequence with large order of magnitude will impact more in these indicators than a sequence with small one. Furthermore, a relative indicator can be meaningful than an absolute one for this problem. Thus, we take logarithm for both true and predicting value to diminish the influence caused by the difference in the order of magnitude, which can be seen in the former part of Eq. 11.

Given the significant learning ability of neural network, however, PRNN may not only capture the real pattens from service invocation sequences, but also capture the noise from them. To avoid overfitting, we apply ℓ_2 norm, a basic regularization technique, on all the weight parameters of fully connected layers, which can be seen in the latter part of Eq. 11.

$$\mathcal{L} = \frac{1}{n} \sum_{(i,t)} \mid \log \frac{y_{it}+1}{\hat{y}_{it}+1} \mid + \sum_k \gamma_k \parallel \mathbf{w}_k \parallel_2 \tag{11}$$

where n is the number of prediction value, $\hat{y}_{it}$ is the prediction value of PRNN for service i in day t, y_{it} is the true invocation records, k is the number of fully connected layers, various γ are coefficients determining the intension of regularization, and various $\mathbf{w}$ are the weight parameters of fully connected layers.

B. Optimization

Since there may be thousands of parameters contained in PRNN, the efficiency and efficacy of parameters learning is non-negligible. To achieve the best results, we adopt RMSprop [8], an extension of stochastic gradient descent algorithm, to optimize the loss function. To implement back-propagation through time (BPTT), we also adopt the gradient clipping method [9] to eliminate the gradient exploding problem.

During the hyper-parameters tuning, we found that 3×10^{-4} can be a good initial learning rate, and updating it by a 0.1 decay every time the validation loss stops decreasing will be helpful to train PRNN efficiently.

Integrating the aforementioned points in Section II and III, Algorithm 1 describes the training process and hyper-parameters settings of PRNN. It is noticeable that PRNN treats its output as input to make prediction when forecasting.

IV. Experiments

In this section, we describe how we tested and verified the efficiency and usability of PRNN on a real-world dataset. Our extensive experiments prove that PRNN reaches a higher prediction accuracy than prior arts.

Algorithm 1 training process of PRNN

Input: service invocation sequences: $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$.
Initialize: learning rate: $lr = 3 \times 10^{-4}$, iteration times: n_{iter}, batch size: $n_{\text{batch}} = 250$, clusters number: $n_{\text{class}} = 14$, service invocation sequences length: n, hidden states number: $n_h = 128$, regularization factor $\gamma = 0.01$.
Output: prediction sequences: $\{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_m\}$.

```
1:  for $i = 1$ to $m$ do
2:      $\mathbf{x}_i \leftarrow \log(\mathbf{x}_i + 1)$
3:  end for
4:  for $i = 1$ to $n_{\text{iter}}$ do
5:      Shuffle the order of service invocation sequences randomly
6:      for $j = 1$ to $\frac{m}{n_{\text{batch}}}$ do
7:          Apply Eq. 8 to calculate the autocorrelation coefficients of service invocation sequences
8:          Apply Eq. 9 to calculate the corresponding probability of service invocation sequences belonging to different clusters
9:          for $t = 1$ to $n$ do
10:             $h_t$ = LSTM($h_{t-1}$)
11:             Apply Eq. 7 to calculate intermediate results $\mathbf{z}$
12:             Apply Eq. 10 to generate $\hat{y}_t$
13:         end for
14:     end for
15:     Apply BPTT to backpropagate gradient
16:     Use $\ell_\infty$ norm to clip gradient
17:     Use RMSprop to set step automatically
18:     Update learning rate
19: end for
20: for $i = 1$ to $m$ do
21:     $\hat{\mathbf{y}}_i \leftarrow \exp(\hat{\mathbf{y}}_i) - 1$
22: end for
```

A. Dataset and Setup

Wikistat[1] is a public dataset, which records millions of Web Page View (PV) from English, Chinese and other wiki-projects, hourly since September 2013. Assume a Web page is viewed as a service, the number of access to the Web page can be viewed as the number of invocations to its counterpart service. For its similarity with the service invocation sequences, therefore, PVs are reasonable to be used to test and verify our technique. In the following experiments, we randomly picked up 29,835 pages from English and Chinese projects, and divided them into two parts. The first one, starting from July 1, 2015 to June 30, 2017 and containing 731 points for each, are used to train the models and make prediction. The other one, starting from July 1, 2017 to August 31, 2017 and containing 62 points for each, are used to evaluate the prediction accuracy and robustness of our models. Numerical properties of the dataset are summarized in Table I.

[1]https://dumps.wikimedia.org

Table I
NUMERICAL PROPERTIES OF DATASET

Item	Number
PVs with decadal order of magnitudes	6,327
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

Fig. 4 demonstrates some typical original PVs in the dataset. The blue line is on behave of some PVs with strong periodicity, which displays a similar tendency when every period comes. The orange line and green line are on behave of such PVs that share similar changing patterns for their similar attributes. From these PVs, we can observe the complicated characteristics and intricate relations between PVs, which is similar with service invocation sequences.

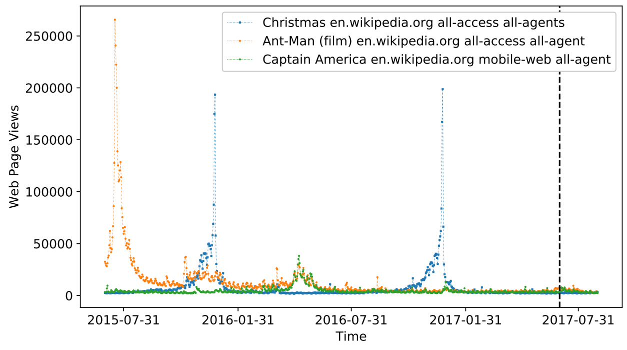


Figure 4. Real page views from Wikistat. For every page, the data before the dashed line are used for training models and making prediction, while the data after the dashed line are for evaluation.

B. Evaluation Scheme

Similar with the consideration of loss function design, we adopt root mean squared logarithmic error (RMSLE) [10] to evaluate different models. Being a metrics reflecting relative predicting error, RMSLE is non-sensitive to the order of magnitude and can be formulated by Eq. 12.

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i,t}\log\left(\frac{y_{it}+1}{\hat{y}_{it}+1}\right)^2} \tag{12}$$

where n is the number of prediction value from wiki pages, $\hat{y}_{it}$ is the prediction value of different models, and y_{it} is the real value of PVs. The value of RMSLE indicates the relative prediction accuracy of PVs by eliminating the order of magnitude between them. It is obvious that a lower

RMSLE value means a higher prediction accuracy. It is also noticeable that RMSLE presents a larger value when underfitting, which is conservative for services providers to maintain the QoS.

C. Benchmarks

To evaluate the performance of PRNN in different views, we carefully chose three baseline methods as below.

1) **Autoregressive Moving Average Model (ARMA)**: ARMA model and Autoregressive Integrated Moving Average model (ARIMA) are classical time series methods, which will treat each service invocation sequence as an individual [5]. Because ARMA (Eq. 13) is designed for stationary time series and ARIMA is for non-stationary one, we first apply Dickey-Fuller test [11] to verify the stationarity of a PV, and then decide which model to apply. A PV is regarded as stationary, if its p–value is less than 0.01; or non-stationary if otherwise. For both models, Bayesian Information Criterion (BIC) is used to select the order of both models. For some PVs whose orders cause singular value decomposition exception in the experiments, we set the last day of training set (June 30, 2017) as the prediction results.

$$x_t = c + \sum_{i=1}^{p} \varphi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}. \qquad (13)$$

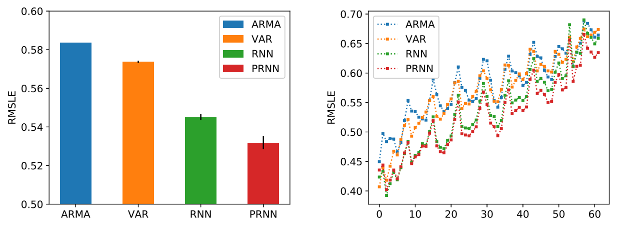
where various φ_p and θ_i are parameters of ARMA, c is a constant, and ε_t is white noise.

2) **Vector Autoregression Model (VAR)**: Vector autoregression is an extension of ARMA model, which is widely used for multivariate time series forecasting [12], [13]. A p–order VAR model can be described as Eq. 14:

$$\mathbf{x}_t = \mathbf{c} + \sum_{i=1}^{p} \mathbf{A}_i \mathbf{x}_{t-i} + \varepsilon_t \qquad (14)$$

where various $\mathbf{A}_i$ are the matrices of coefficients, $\mathbf{x}$ is the vector of past sequences, ε is zero-mean white noise, and $t = k+1, \ldots, T$, where T denotes the length of time series.

3) **Recurrent Neural Networks (RNN)**: Recurrent neural networks is one of the best models for sequences. Particularly, Long Short-Term Memory (LSTM), a special structure of RNNs, is excellent in capturing long-term dependency of sequences [14]. As a degenerative model of PRNN (see Section II-B), we trained and predicted PVs by setting all parameters of RNN the same as that of PRNN.



(a) Total RMSLE of four models (b) RMSLE with growing periods

Figure 5. RMSLE of PRNN, RNN, VAR and ARMA of repeated trials.

D. Quantitative Comparisons

To test and verify the validity and robustness of PRNN, we designed and conducted 10 experiments with shuffled data repeatedly, with the baselines methods as comparison.

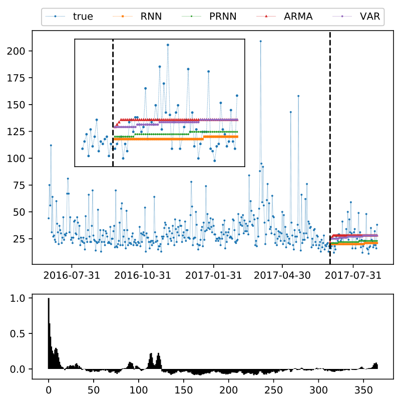
Fig. 5 (a) demonstrates the RMSLE of PRNN and baselines methods with two months (62 days) prediction. The average RMSLE of ARMA, VAR, RNN and PRNN are 0.5836, 0.5737, 0.5451 and 0.5319, respectively. We observe that RNN and PRNN are markedly better than other methods, due to the effective feature extraction by LSTM. Specially, PRNN reduces the RMSLE by 2.48% than RNN in terms of prediction accuracy, which is benefited from the piecewise mechanism. The standard deviation of ARMA, VAR, RNN and PRNN are 0, 0.0006, 0.0015 and 0.0033, respectively. The standard deviation of ARMA is the lowest, because once the order of ARMA is determined by the algorithm, PVs will be modeled by the same parameters. The standard deviations of other models are also low enough to be acceptable, showing their robustness.

Fig. 5 (b) shows the RMSLE of PRNN and the baselines methods with the increment of prediction days. In the figure, RMSLE of four methods are close and relatively low in the first five prediction days. With the increment of prediction days, RMSLE fluctuates and presents an increasing tendency. In particular, compared with the baseline methods, PRNN tends to present a far better prediction accuracy as the prediction days increase.

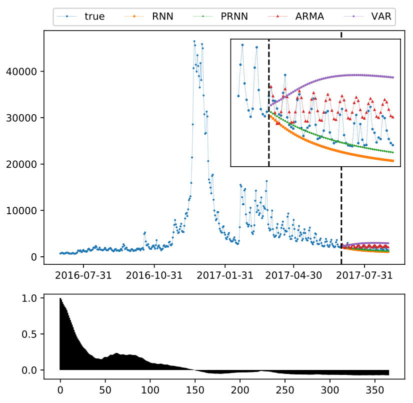
E. Case Study

To analyze the performance of these models, we observed lots of prediction results and plotted some typical cases in Fig. 6, where the upper figures demonstrate the true and predicting values of three different PVs respectively and the lower ones represent their autocorrelation coefficients. Particularly, the blue line represents the true values, while other colors after the dashed line are predicting ones. Particularly, the red, purple, orange and green lines represent the prediction results of ARMA, VAR, RNN and PRNN, respectively.

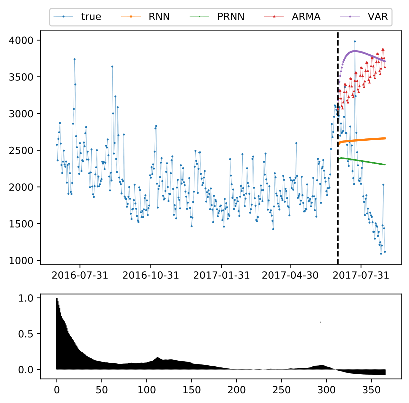
Fig. 6 (a) is the typical representative of such PVs, whose autocorrelation coefficients concentrate on zero, which can be regarded as noise sequences. In the real case *Doomsday*



(a) *Doomsday (comics) en.wikipedia.org*



(b) *Assassin's Creed (film) en.wikipedia.org*



(c) *Ant-Man (film) en.wikipedia.org*

Figure 6. Above are some typical cases of experiments results. Blue line is the true PV of a wiki–page, and red, purple, orange and green lines represent the prediction results of ARMA, VAR, RNN and PRNN, respectively. Before the dashed line (July 1, 2017) is a year of real data for training models. Below are 365 days autocorrelation, which is calculated by training data, corresponded to the above PV.

(comics) en.wikipedia.org all-access spider, all the methods present similar prediction tendency, centralizing near the average of the sequence. The RMSLE of ARMA, VAR, RNN and PRNN are 0.3709, 0.3623, 0.3819 and 0.3535, respectively, showing that PRNN is the best in this situation. In other analogous cases, PRNN also predicts the tendency of services invocation near the average value. Although the prediction ability of PRNN and the baseline methods are close in this situation, sufficient experiments show that PRNN has a narrow advantage than other methods in most cases.

Fig. 6 (b) and (c) are some typical representatives of a set of PVs, which have strong relevance with recent days and barely have relevance with the past. Taking a real PV *Assassin's Creed (film) en.wikipedia.org mobile-web all-agents* as an example, the prediction tendency of ARMA is similar with the real PV, while prediction tendency of RNN and PRNN present a major decreasing trend. However, the RMSLE of ARMA, VAR, RNN and PRNN are 0.2540, 0.5023, 0.3454 and 0.2301, respectively, showing the powerful predicting ability of PRNN. In another real case *Ant-Man (film) en.wikipedia.org desktop all-agents*, ARMA predicts an opposite tendency of the PV mistakenly, which is attributed to an upward trend before prediction and the limitation of ARMA to capture the long-term dependency. However, RNN and PRNN predict the PV in the right trend benefited from long-term dependency features extracted by LSTM units. The RMSLE of ARMA, VAR, RNN and PRNN are 0.6547, 0.7083, 0.4331 and 0.3442, respectively. From these cases, we hold that ARMA does able to capture the tendency of the sequence, but it is also likely to predict an opposite trend. On the contrary, PRNN is more stable in terms of RMSLE, although it only captures the baseband tendency in sequences.

From the aforementioned cases, we can find that VAR models are always influenced by other PVs in the same group, so its performance is week and unstable. The prediction results of ARMA is easily affected by the real PVs near the prediction boundary. The prediction results of PRNN is better than RNN in most cases due to our piecewise mechanism. Particularly, PRNN performs much better than the baseline methods with prediction days increasing.

V. Related Work

With the development of service ecosystem, predicting the tendency of services invocation is becoming increasingly important. Since service invocation records are sequences, time series prediction models can be a kind of solutions for the problem.

To predict the tendency of services invocation, some main characteristics, such as linearity, nonlinearity, periodicity and long-term dependency, should be paid great attention to. Among prior arts, Autoregressive Moving Average Model (ARMA) for stationary time series and Autoregressive Integrated Moving Average Model (ARIMA) for non-stationary time series have been designed to capture the linearity of time series, thus are acknowledged to be powerful for time series prediction [5]. However, these linear models fail to seize the nonlinearity and long-term dependency of time series. Ding et al. have developed support vector machine to regress (SVR) the nonlinearity of time series, which makes up for the limitation of linear models [15]. Since the appearance of LSTM, long-term dependency of time series have been captured.

Besides the complicated characteristics, the promising relations between service invocation sequences are also non-negligible. To utilize such relations to improve the prediction accuracy, Vector Autoregression (VAR) model, an extension of ARMA, models various time series as vector, building the connection between time series and being a better solution [12]. Similar with ARMA, VAR can only capture the linearity of time series. Thus, Bao et al. developed SVR for multi-variable time series prediction to capture the nonlinearity [16]. However, if the amount of time series is too great, these methods will be too time-consuming or memory-consuming to lose their efficacy. In recent years, the advancement of neural network supplies another choice to leverage such relations. Lee et al. used RBF neural network to predict time series [17]. Dasgupta and Osogami proposed Nonlinear Dynamic Boltzmann Machines to predict time series [18]. Further, recent researches have shown that RNN seems to be more effective in modeling time-series data, and in particular, LSTM-based RNNs, being able to capture the long-term dependency, are better in predicting the tendency of services invocation [7], [19]. Compared to the related work, we apply LSTM to extract general features of service invocation sequences, and develop a piecewise instrument to categorize service invocation sequences into clusters thus to take into consideration of their discriminations.

VI. Conclusions

In this paper, we have presented a novel deep neural network model named Piecewise Recurrent Neural Networks (PRNN) to predict the tendency of services invocation. The main ideas include: 1) applying LSTM units to model the complicated characteristics of service invocation sequences generally; and 2) developing a piecewise mechanism to predict different types of service invocation sequences discriminatingly. Extensive experiments have proved that PRNN performs better than the baseline methods in prediction accuracy.

Our future work will focus on two aspects: 1) make a lucubrate research on service invocation sequences to further improve the prediction accuracy; and 2) predict the tendency of services invocation in units of hours.

Acknowledgment

This research has been partially supported by the National Nature Science Foundation of China (No.61673230).

References

[1] S. Chen, Y. Fan, W. Tan, J. Zhang, B. Bai, and Z. Gao, "Time-aware collaborative poisson factorization for service recommendation," in *IEEE International Conference on Web Services*, 2016, pp. 196–203.

[2] 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.

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[4] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

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

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] 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.

[8] T. Tieleman and G. E. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," 2012.

[9] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Computer Science*, vol. 52, no. 3, pp. III–1310, 2012.

[10] S. Jachner, K. G. V. D. Boogaart, and T. Petzoldt, "Statistical methods for the qualitative assessment of dynamic models with time delay (r package qualv)," *Journal of Statistical Software*, vol. 22, no. 8, pp. 1–30, 2007.

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

[12] H. Ltkepohl, "New introduction to multiple time series analysis," *Economic Record*, vol. 83, no. 260, p. 109110, 2007.

[13] M. Pompella, *Analysis of Financial Time Series*. Wiley, 2005.

[14] D. Wierstra and F. Gomez, "Evolino: hybrid neuroevolution / optimal linear search for sequence learning," in *International Joint Conference on Artificial Intelligence*, 2005, pp. 853–858.

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

[16] 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.

[17] 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.

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

[19] D. Hsu, "Time series forecasting based on augmented long short-term memory," 2017.