Web service workload prediction using deep learning*

†

Stefan Sebastian‡
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, State1, Romania
sshp1629@scs.ubbcluj.ro

Virginia Niculescu§
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, State1, Romania
vniculescu@cs.ubbcluj.ro

Abstract

Web services....

The paper presents....

Keywords web services, workload, prediction, microservice capacity, performance modeling microservice, microservice applications

1 Introduction

Web services....

Micro services

Workload evaluation is very important .. in order to assure efficient ...

The main goal of the presented investigation was to find a performant model for web application workload prediction, which can be later used by a proactive micro-service scalier.

The investigation uses different architectures of deep learning: MLP, CNN, CNN-LSTM hybrid.

The main contribution of this research is the application of deep learning to this specific problem and the comparison with a classic timeseries approach (ARIMA).

The paper is structured as follows:

Section **??** succinctly describes

Section **??** presents a

Section **??** presents

Conclusions and future work are presented in section **??**.

*with title note
†with subtitle note
‡with author1 note
§with author1 note

ESEC/FSE 2019, ,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 Related work

Calheiros et al.[**?**] apply the ARIMA model to cloud workload prediction. The model was evaluated using a trace of English Wikipedia resource requests spanning a duration of four weeks. The first three are used for training and the fourth for prediction using a time window of 1 hour. The MAPE varies from 9% to 22% depending on the confidence interval chosen, from 80 to 95 which is meant to limit the occurences of underestimations.

Other classic timeseries models have also been applied for this task, like Brown Exponential Smoothing by Mi et al.[**?**] obtaining a Mean Relative Error of 0.064 on the France World Cup 1998 web server trace. Another classic model is Weighted Moving Average, applied by Aslanpour et al.[**?**] in which recent observations are given more weight based on the Fibonacci rule, and was tested on a NASA server 24h trace achieving a 5% improvement in response time on a cloud scaling simulator.

Kumar and Singh[**?**] applied artificial neural networks for workload prediction on a seven month log of traffic from a Saskatchewan University web server and a two month one from the NASA Kennedy Space Center web server. They use a classic ANN architecture : one input layer(size 10), one hidden and one output, and the model is trained through the SaDE technique, which means learning its weight through evolutionary algorithms. The results of this model were compared to an ANN trained through backpropagation: 0.013 and 0.001 for D1 and D2 vs 0.265 and 0.119, using the RMSE metric over normalized data.

CloudInsight[**?**] is one of the most complex models for workload prediction. It uses a technique called "council of experts", meaning an ensemble of different models, in this case: classic timeseries (autoregressive, moving average, exponential smoothing), linear regression, and machine learning (SVM). Each model has a different prediction weight which is also learned real-time through a SVM based on their accuracy on the dataset. The evaluation was done on a subset of the wikipedia trace[**?**], on google cloud data and on some generated workloads. An indicator of performance is normalized RMSE, meaning how much better it performs than other models. On average it was 13% to 27% better than baselines (ARIMA, FFT, SVM, RSLR).

A review of how deep learning methods can be applied to time series problems was created by Gamboa[?]. The paper distinguishes between three types of problems: classification, forecasting and anomaly detection, presents methods for modelling them and guidance for selecting appropiate models. It also shows an improvement in performance over the previously existing techniques. Brownlee[?] published a comprehensive guide on applying MLPs, CNNs and LSTMs on various real datasets and discussed their advantages over classic methods, which were used as baselines for the experiments.

Lin et al[?] proposed a hybrid CNN LSTM architecture for learning trend in time series. It relies on on CNN to extract important features from raw timeseries data and LSTM to find long range dependencies in historical data. The model was shown to outperform both CNN and LSTM with around 30% lower RMSE on 3 real world datasets.

3 Web services

TO BE ADDED

architecture

[?]

4 Methodology

[?]

The methodology for tuning the models is based on the following main stages:

- choose a realistic data set...
- prepare the data such that ...
- apply baseline models
- apply CNN models
- compare and analyse de results

Data preparation

In order to turn a web request log file into a supervised dataset the following steps were, taken:

- create buckets which contain the number of requests in a time interval,
- iterate over the buckets using the sliding window technique [1].

Basically we generate training instances with input (t, t-1, ... t-n) and output (t+2). The predicted value is t+2 instead of t+1 because a scaler using this model would need to have a buffer window during which to deploy the services.

4.1 Performance Metrics

The error metrics selected for this experiment are: Mean squared error, $MSE = 1/n \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$, Mean absolute error, $MAE = 1/n \sum_{i=1}^{n} |Y_i - \hat{Y}_i|$, and Mean absolute percentage error, $MAPE = 1/n \sum_{i=1}^{n} (Y_i - \hat{Y}_i)/Y_i$, as described in [?]. MSE was used as the loss function for training because it tends to penalize big deviations in prediction, which is desirable for our problem as we want to accurately predict traffic spikes. MAE is similar, but conceptually simpler, given that each prediction error contributes in proportion to its absolute value.

MAPE is independent of the problem scale and can be interpreted intuitively, therefore can be used to give a general impression of how well a model performs across diferrent datasets. According to [?] a highly accurate forecast would have MAPE lower 10%, and a good forecast between 10 and 20%.

4.2 Baseline models

Two baseline models were applied: the naive approach (predict traffic in window t+2 to be traffic in t) and a classic timeseries model (ARIMA).

ARIMA [4], which stands for autoregressive integrated moving average, is a classic approach to modelling timeseries. In order to apply this model we need to

nd appropiate values for its parameters: p, q, d. The value of d means the number of times the series needs to be differentiated in order to make it stationary.

The series stationarity was checked using the augmented Dickey- Fuller test [2] which found the p-value to be 1.0902496274664773e -??? experiments????

These baseline models have been chosen since they are considered....

4.3 Deep Learning Models

We have chosen in this investigation to apply the following deep learning architectures:MLP, CNN, CNN-LSTM hybrid details.

These architectures were chosen for their characteristics and ... previous work

4.3.1 MLP

4.3.2 CNN

4.3.3 CNN-LSTM Hybrid

This model was applied on a range of timeseries tasks by Lin et al. [9]. It relies on LSTM to find long range dependencies and historical trends and on CNN to extract important features from raw timeseries data.

The starting values for some parameters were influenced by the research done by Lin et al. [9]: 32 cnn filters, 1 lstm layer with a couple hundred units.

5 Experiments and Evaluation

In this section we will described the experiments that we conducted, their results and a comparative analysis of these results.

We have chosen for the *Japanese wikipedia trace ????* because besides that fact that includes significant patterns, it also has some interesting irregularities, like a huge spike which is not repeated.

Figure 1. The results of ARIMA .

Figure 2. Autocorrelation plot.

A realistic workload has been used – a wikipedia trace for 12 days in september 2007. From this data set, a subset of requests was extracted (all requests for Japanese wikipedia). The subset data was also used by Kim et al. [**?**] and we intend to compare to obtained results.

5.1 Settings

The selected time window for tuning is 10 min. (todo test on other windows - tune on other windows?)

The dataset is split into a training and testing with a ratio of 0.9.

The validation method is k-fold Cross-Validation [10] with k = 3, which means splitting the training dataset into k equal parts, perform training on k - 1 and evaluation on the part left out.

This process is repeated k times.

In order to avoid the model to be dynamically influenced, we do not modify the testing data during the model tuning.

5.2 Naive baseline

5.3 ARIMA

Settings

Results

The series stationarity was checked using the augmented Dickey- Fuller test [2] which found the p-value to be 1.0902496274664773e-08. This is lower than 0.05, the commonly used threshold, meaning we can set d to 0. The partial auto-correlation plot was analyzed to set the auto-regression parameter (p). From Figure 2 we can see that the significance region is confidently passed at 1, with a steep decline afterwards. The moving average parameter (q) is approximated from the auto-correlation plot. It suggests a value of around 20 would be a good start. After fitting ARIMA(1,0,20) the final 2 layers had P-value of 0.547 and 0.758 which meant that they were not significant. After trying some values for q: 5,10,15,18 the best results were obtained on ARIMA(1,0,15) with MSE: 15289199.210, MAE: 3118.982, MAPE: 6.782.

5.4 MLP

Settings

Results

After some manual experiments started with a MLP with 2 hidden layers (150, 100 neurons) and sliding window size of 24 (input size). To find an optimal combination of batch size and epoch no a 2d grid search was performed 3. Batch size should ideally be a power of 2 for extra performance on GPU architectures, as some experiments were ran on Google Colab. Lower batch size is more accurate but training is slower [6]. As expected the best MSE is obtained for the lowest batch size(4) however it does not drop significantly at 8, regardless of epochs no. The selection of epoch no is again a trade-off between speed and accuracy. We see a smaller no of epochs(50) performs poorly, while the difference between 100 and 250 is not that great, meaning that we can get a good approximation of a model using a batch size of 100. Some experiments were done with adding Dropout layers on different values (0.2, 0.1, 0.05) however it did not improve performance. These are generally used to prevent overfitting, when the network is too big, the data is scarce or training is done for too long [11], which was not the case for this experiment. Various optimizer and activation functions were tested. The Adadelta optimizer and the relu activation were selected. A comprehensive grid search was performed for sliding window size and number and content of hidden layers, of around 90 combinations. Some of the best performing are presented in table 2.

5.5 CNN

Settings

Results

5.6 CNN-LSTM Hybrid

The starting values for some parameters were influenced by the research done by Lin et al. [9]: 32 cnn filters, 1 lstm layer with a couple hundred units.

The results are presented in Figure **??**

5.7 Analysis

final analysis of the results

Table 1. Final results

DataSet	Naive	ARIMA	MLP	CNN	CNN-LSTM
Jp10	20233320	15289199	8591086	12766376	7252806
Jp15	87883950	56662039	31042348	36278866	todo
De10	16789198	10318406	5180340	todo	todo
De15	77481580	43398336	17276322	todo	todo

6 Conclusions

The paper analyzes