Proceedings of the Federated Conference on Computer Science and Information Systems pp. 141–144
DOI: 10.15439/2020F101
ISSN 2300-5963 ACSIS, Vol. 21

A Framework for Time Series Preprocessing and History-based Forecasting Method Recommendation

Marwin Züfle, Samuel Kounev
University of Wuerzburg, Germany
Email: {marwin.zuefle, samuel.kounev}@uni-wuerzburg.de

***Abstract*—The complexity of managing the capacities of large IT infrastructures is constantly increasing as more network devices are connected. This task can no longer be performed manually, so the system must be monitored at runtime and estimations of future conditions must be made automatically. However, since using a single forecasting method typically performs poorly, this paper presents a framework for forecasting univariate network device workload traces using multiple forecasting methods. First, the time series are preprocessed by imputing missing data and removing anomalies. Then, different features are derived from the univariate time series, depending on the type of forecasting method. In addition, a recommendation approach for selecting the most suitable forecasting method from this set of algorithms for each time series based only on its historical values is proposed. For this purpose, the performance of the forecasting methods is approximated using the historical data of the respective time series under consideration. The framework is used in the FedCSIS 2020 Challenge and shows good forecasting quality with an average R^2 score of 0.2575 on the small test data set.**

I. Introduction

OVER the last decades, the network load in large IT systems has grown considerably. Thus, coping with the increased data traffic is becoming more and more difficult. Typical reactive mechanisms that adapt the system to the current condition are no longer applicable, as this leads to temporary overload situations with resulting delays. To overcome this problem, proactive adaptation algorithms are required that analyze historical data and automatically forecast future conditions to enable early decision making. However, the decision making component is beyond the scope of this paper, as this paper is part of the *FedCSIS 2020 Network Device Workload Prediction Challenge* [1]. To achieve sufficient forecasting performance, no single method can be used since the "No-Free-Lunch-Theorem" states that there cannot be a single algorithm that outperforms all others on every kind of data [2]. For this reason, we developed a hybrid approach that recommends the best forecasting method for a given time series based only on its known historical values. In addition, we introduce an algorithm for missing data imputation and a technique for eliminating anomalies to preprocess the time series in advance.

The remainder of this paper is structured as follows: In Section II, we present related work on time series forecasting. The foundations of the applied forecasting methods are described in Section III. In Section IV, we introduce the preprocessing steps that were applied prior to the modelling part of the approach (Section V). Experimental results are presented in Section VI. Finally, Section VII concludes the paper.

II. Related Work

Forecasting time series is a widely studied field of research for which many different approaches exist. Firstly, individual methods can be used to forecast time series. This area ranges from the application of statistical methods [3] to machine learning models [4]. However, according to the "No-Free-Lunch-Theorem", there is no single method that surpasses all other methods for every type of data [2]. Therefore, more sophisticated approaches implement hybrid forecasting methods. That is, several individual forecasting methods are applied and the final result is either a weighted combination [5], [6], a sequential execution of methods on different parts of the time series [7], [8], or the forecast of a recommended method.

First approaches towards forecasting method recommendation use manually created expert systems [9]. One of the first works using automatic rule induction methods is by Arinze et al. [10]. More recent approaches to the recommendation of forecasting methods are by Wang et al. [11] and Züfle et al. [12]. However, all of these automatic rule learning approaches calculate characteristics of the time series in a large training data set and assess the forecasting accuracy of the available methods on them. Then, a rule induction technique is applied to map the characteristics of the time series to the best performing forecasting method. Therefore, these approaches require a large training data set that equally covers all time series characteristics. In contrast, the approach presented in this paper does not require such a large training data set. Instead, the performance of the different forecasting methods is estimated on a part of the time series to be forecast. Furthermore, no rule learning approach is required because our framework selects the forecasting method with the highest R^2 score on the validation part of the considered time series.

III. Background

In this section, the applied forecasting methods and the FedCSIS 2020 Challenge data set are briefly presented.

A. Time Series Forecasting Methods

In this paper, we apply six different forecasting methods from three different categories. The first category consists of two simple statistical features, i.e., median and mode. For time series with very little information content, forecasting these constant values can achieve a better accuracy than using more sophisticated forecasting methods. The second category are machine learning methods that require derived features

as input. Here, we apply two regression techniques, i.e., Random Forest [13] and XGBoost [14]. We also use Random Forest in classification mode when the time series meets a certain requirement. In some previous works, we have already developed a novel forecasting method for seasonal, univariate time series [15], [16]. This method is called *Telescope* and is available on our GitHub repository[1]. We use Telescope in two alternative ways, which forms our third category of forecasting methods. Telescope does not require feature generation by the user. Instead, Telescope includes an internal feature generation mechanism. First, Telescope estimates the frequency of the seasonal pattern and removes anomalies in an internal preprocessing step. Next, Telescope generates features by splitting the univariate time series into seasonal, trend, and residual components. Here, a heuristic is implemented that estimates whether the time series exhibits an additive or multiplicative composition. If the composition is multiplicative, a logarithm is applied to the time series to transform the composition into additive mode. Each of the components is then forecast separately. The fine-grained seasonal pattern is continued, since the definition of seasonality states that the seasonal pattern must not change. In addition, categorical information is extracted and forecast using an artificial neural network. The trend is predicted using an ARIMA model. Finally, the categorical information, the seasonal forecast, and the trend forecast are passed to XGBoost, which recombines the forecasts and predicts the residual component. For more details on how this forecasting method works, see [15] and [16].

B. Challenge Data Set

The data set of the FedCSIS 2020 Challenge consists of network device workload traces. Each trace consists of the target variable *Mean*, the corresponding *time_window* in hourly resolution, and up to eight additional features captured between 2 December 2019 and 20 February 2020. Thus, each feature of a trace consists of a maximum of 1924 monitoring entries. However, the traces also contain missing data. These data gaps range from individual entries up to several hours. Finally, the goal of the challenge was to forecast the *Mean* one week in advance, i.e., 168 values, for 10,000 time series.

IV. Preprocessing

As our approach relies on time series forecasting, we did not go deeper into the features other than *time_window*, nor did we examine time series other than those needed to be forecast.

A. Missing Data Imputation

After analyzing the data, we found that most of the missing values are at the beginning of the time series or that only a few consecutive data points are missing. We did not reconstruct missing values at the beginning of a time series, since these gaps can extend to several hundred values. The reconstruction of such long data series is typically highly error-prone and would therefore worsen our model. In addition, missing data at the beginning is not critical, since it merely shortens the time series. To impute the missing values within the time series, we assume a daily pattern within the data. Since the data are aggregated hourly, we set this seasonal time offset to 24. In addition to the daily pattern, we analyze whether there is a trend between the day of the missing data and the next or previous day. Then, the algorithm implants the missing value by multiplying, respectively dividing, the known value one season before, respectively after, the missing value by the derived trend factor. We apply this procedure in chronological order so that imputed values can be used to impute subsequent missing values. In case that there are still few missing values after applying this method (i.e., the values one season before or after the missing value are also missing), the value is imputed by linear interpolation between the last known value before and the first known value after the missing value.

B. Anomaly Removal

While analyzing the time series, we also saw that some time series had high spikes. Since these outliers worsen the learned models, we apply a method for detecting anomalies. Here, we use a modified version of the well-known "three-sigma rule". In contrast to the typical three sigma rule, we use the median instead of the mean value as a baseline, since the distributions of the time series are not necessarily symmetrical. Furthermore, we calculate the standard deviation only between the 1st and 99th percentile of the data, since potential outliers would already influence the standard deviation if it was calculated over the entire time series. We also set the tolerance multiplier to a more conservative value (i.e., 10-sigma rule) because we do not want to remove the normal peaks in a daily workload pattern. After detecting outliers, the algorithm overwrites these values with a linear interpolation between the non-anomalous predecessor and the non-anomalous successor.

V. Modelling

After imputing missing values and removing outliers, the main part, i.e., the modelling, takes place. Fig. 1 shows the simplified overall workflow of our approach.

A. Frequency Estimation

In the modelling part, we first estimate the seasonal frequency of the time series. Telescope already provides such a frequency estimation method, which uses a periodogram to extract the most dominant frequencies and searches for meaningful human-based frequencies nearby. Here, we have limited the possible results of the frequency estimation method to -1 (no frequency found), 24 (daily), and 168 (weekly).

B. Feature Generation

Afterwards, the lags of the univariate time series are generated. We used the lags one to six for all time series and if the time series has a seasonal pattern (i.e., the frequency is 24 or 168), we also added the lags 24 and 168 to provide not only the most recent data as features for the machine learning models, but also those from a day ago and a week ago.

In addition to the delayed time series, we also provide the hour of the day, the day of the week, and whether the day is

[1]GitHub link to Telescope: https://github.com/DescartesResearch/telescope

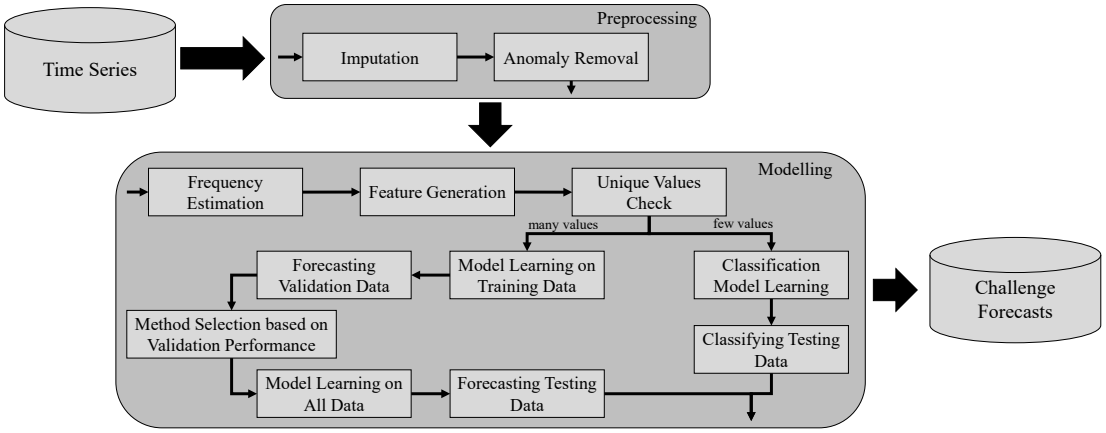

Fig. 1. The overall workflow of the framework including preprocessing and modelling.

a holiday or not as features for the machine learning models. These features are required as they can contain additional seasonal information or explain deviations from normal behavior.

C. Classification

The algorithm then determines the number of unique values within the time series. We have found that the data set contains several time series with only a few different values and, most importantly, no trend pattern. In such cases, classification can be advantageous over regression models. Thus, if we observe that a time series consists of less than six different values, we learn a random forest [13] multi-class classification model with each class representing the corresponding value.

In the specific case that the time series consists of only a single value, we predict exactly this value since the available training data does not contain any further information.

D. Regression Method Recommendation

In contrast, we apply six different regression methods when we find six or more different values in a time series ("No-Free-Lunch-Theorem"). For this purpose, we have again split the training data into a training and a validation set. The validation set consists of the last 168 values, while the training part contains all previous values. In the following, we use the term training data for this subset of the FedCSIS 2020 Challenge training data and validation data for this horizon within the FedCSIS 2020 Challenge training data. For the test data, we still refer to the unknown data that was used by the creators of the FedCSIS 2020 Challenge for the final evaluation.

The used methods are median, mode, Telescope with and without enabled frequency estimation, Random Forest, and XGBoost. The first two methods forecast the median, respectively the mode, of the training part for the entire horizon. Both versions of Telescope learn internal features but do not use the features created above, while Random Forest and XGBoost only get the lagged time series, hour of day, day of the week, and holiday as features. For both machine learning methods, we have carried out a hyper-parameter optimization. Since we predict 168 values at once, we have to fill our lag features during runtime by starting with the original values given by the training data and gradually filling them with our forecasts. That is, we forecast each value in the horizon as a one-step-ahead forecast, and after each of these one-step-ahead forecasts, we must create the feature set for the next value. However, the model remains the same, only the feature set must be recreated for each value in the forecast horizon.

To estimate the best method for a given time series, we use the validation data to calculate the R^2 score of each method. Then, we select the method that achieved the highest R^2 score and learn a new model using the entire time series (i.e., training and validation data). Finally, we forecast the 168 values using the presumably best forecasting method and adjust the forecasts under the assumption that the data should not contain negative values. Therefore, we set negative forecasts to zero if all values in the training data are non-negative. There are only few time series with negative values in the training data. For these time series, we set the forecasts that are smaller than the minimum in the training data to exactly this minimum as we interpret it as a kind of zero-baseline.

VI. Experimental Results

This section presents the experimental results of our framework based on the FedCSIS 2020 Challenge data set. First, Fig. 2 shows a time series with a gap of seven missing values. Here, the original time series is depicted in black, while the green color indicates the imputation generated by our algorithm. It can be seen that the imputation creates reasonable reconstructions. In particular, the imputation algorithm even reconstructs a first spike for the double-spiked seasonal pattern, similar to the other seasonal highs, because the algorithm considers precursors and successors with a distance equal to the frequency of the seasonal pattern.

Fig. 3 shows an exemplary anomaly removal in one of the competition time series. The black line shows the corrected time series, while the red line shows the anomalous values from the original time series. It can be seen that the peak value of the daily pattern significantly exceeds the normal range and

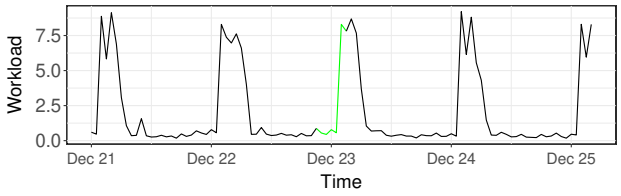

Fig. 2. An example time series with imputed values.

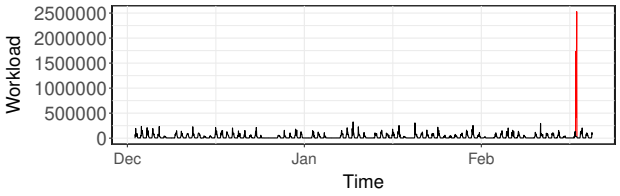

Fig. 3. An example time series with removed anomalies.

our anomaly detection method therefore overwrites these values by interpolating between the first non-anomalous precursor and the next non-anomalous successor. If such anomalies were not removed before modelling, the forecasting methods could learn a different, incorrect behavior. In particular, if the anomaly is at the end of a time series, as shown in Fig. 3, the trend component can be manipulated so that the approximation would erroneously detect an exponential trend.

The measure of the FedCSIS 2020 Challenge is the R^2 value with f_i and y_i representing the forecast and real value at time i, respectively, while $\overline{y}$ is the mean value of the time series:

$$R^2(f, y) = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \overline{y})^2}$$

The following results are based only on the small test data set of the FedCSIS 2020 Challenge. Using only single forecasting methods with the features explained in Section V, XGBoost yielded the best results with an R^2 score of -0.0072. By adding mode, median, and Random Forest regression together with the recommendation strategy, the R^2 score rose to 0.2012. After including both versions of Telescope into the set of possible forecasting methods, our framework achieved an R^2 score of 0.2544. Finally, by using Random Forest classification for time series with only a few different values, we achieved our highest R^2 score of 0.2575. Since the baseline has an R^2 score of 0.2267, our last two versions clearly surpass the baseline on the small test data set.

The distribution of forecasting methods recommended by our framework is as follows: 124 time series show no variation and therefore, the constant value is forecast. Random Forest classification is applied for 104 time series that have more than one and less than six individual values. For the remaining 9772 time series, regression is used. Mode and median are used 593 and 768 times, respectively. Although XGBoost performed best as a single method, it is only used 1510 times for the entire data set, while Random Forest regression is used most often, i.e., 3280 times. Both Telescope alternatives are applied almost equally often. Telescope without internal frequency estimation is used for 1809 time series, while the Telescope with internal frequency estimation is used for 1812 time series.

VII. Conclusion

In this paper, we introduced our approach used for the FedCSIS 2020 Data Mining Challenge. First, we imputed the time series as they contained missing values and removed anomalous peaks. To tackle the "No-Free-Lunch-Theorem", our approach uses the corrected data to learn several models, from median and mode to univariate time series forecasting and machine learning models with lags and time information as features. Furthermore, our approach applies a recommendation to estimate the best of these methods based on the training performance for each time series. For time series with only a few different values, we apply Random Forest classification instead of regression. For the small testing set, we obtained an R^2 score of 0.2575, which clearly exceeds the baseline.

References

[1] A. Janusz, M. Przyborowski *et al.*, "Network Device Workload Prediction: A Data Mining Challenge at Knowledge Pit," in *Proceedings of FedCSIS 2020, Sofia, Bulgaria*, 2020.
[2] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. on Evol. Computation*, vol. 1, no. 1, 1997. doi: 10.1109/4235.585893
[3] R. N. Calheiros, E. Masoumi *et al.*, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE Trans. on Cloud Computing*, vol. 3, no. 4, 2014. doi: 10.1109/tcc.2014.2350475
[4] K. Cetinski and M. B. Juric, "Ame-wpc: Advanced model for efficient workload prediction in the cloud," *Journal of Network and Computer Applications*, vol. 55, 2015. doi: 10.1016/j.jnca.2015.06.001
[5] J. M. Bates and C. W. Granger, "The combination of forecasts," *Journal of the Oper. Res. Society*, vol. 20, no. 4, 1969. doi: 10.2307/3008764
[6] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *Int. Journal of Forecasting*, vol. 5, no. 4, 1989. doi: 10.1016/0169-2070(89)90012-5
[7] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, 2003. doi: 10.1016/s0925-2312(01)00702-0
[8] N. Liu, Q. Tang *et al.*, "A hybrid forecasting model with parameter optimization for short-term load forecasting of micro-grids," *Applied Energy*, vol. 129, 2014. doi: 10.1016/j.apenergy.2014.05.023
[9] F. Collopy and J. S. Armstrong, "Rule-based forecasting: Development and validation of an expert systems approach to combining time series extrapolations," *Management Science*, vol. 38, no. 10, 1992. doi: 10.1287/mnsc.38.10.1394
[10] B. Arinze, S.-L. Kim, and M. Anandarajan, "Combining and selecting forecasting models using rule based induction," *Comp. & Oper. Research*, vol. 24, no. 5, 1997. doi: 10.1016/s0305-0548(96)00062-7
[11] X. Wang, K. Smith-Miles, and R. Hyndman, "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series," *Neurocomputing*, vol. 72, no. 10-12, 2009. doi: 10.1016/j.neucom.2008.10.017
[12] M. Züfle, A. Bauer *et al.*, "Autonomic forecasting method selection: examination and ways ahead," in *Proceedings of ICAC 2019*. IEEE, 2019. doi: 10.1109/icac.2019.00028
[13] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, 2001.
[14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of SIGKDD 2016*, 2016. doi: 10.1145/2939672.2939785
[15] M. Züfle, A. Bauer *et al.*, "Telescope: A Hybrid Forecast Method for Univariate Time Series," in *Proceedings of ITISE 2017*, September 2017.
[16] A. Bauer, M. Züfle *et al.*, "Telescope: An automatic feature extraction and transformation approach for time series forecasting on a level-playing field," in *Proceedings of ICDE 2020*, April 2020. doi: 10.1109/icde48307.2020.00199