Distributed Real-time Forecasting Framework for IoT Network and Service Management

Diogo Ferreira
Instituto de Telecomunicações
DETI - University of Aveiro
Portugal

Carlos Senna
Instituto de Telecomunicações
Aveiro
Portugal

Susana Sargento
Instituto de Telecomunicações
DETI - University of Aveiro
Portugal

***Abstract*—For efficient network management, it is important to monitor and analyse the data, particularly big data applications based on time series, in terms of trends and correlations, to predict network problems and be able to react preventively. Machine learning techniques can help but, given the amount and complexity of the algorithms and metrics available, the use of these techniques is laborious and requires specialized knowledge. This paper proposes a framework for distributed real-time time series forecasting with the goal to make predictions for various dynamic systems simultaneously and provide straightforward horizontal scaling, increased modularity, high robustness and a simple interface for users. Moreover, the proposed framework also enables the creation of ensemble algorithms, combining the results of multiple predictors, without changes on each individual predictor component. To demonstrate the functionalities of our framework, we show how simultaneous predictions can be made about the number of Internet sessions, using a real data stream from users of the buses in the Porto city.**

***Index Terms*—Time series prediction, Forecasting, 5G, Network Monitoring, Network Management, Machine Learning.**

I. Introduction

In the last few years, deep learning architectures have shown the capability to predict the expected activity of the dynamic systems, predict anomalies or learn new behaviours of these systems in real-time. However, it is still hard to make efficient predictions using machine learning algorithms due to the complexity associated with the configuration and the training of the models. This paper proposes a framework that makes the prediction task of multiple time series systems easier, being able to make predictions in real-time using a microservices' architecture. Moreover, it is able to build an ensemble predictor that performs better than the individual prediction components. The predictions are made through an uniform interface that simplifies the implementation details of the various predictors.

To demonstrate the proposed framework, we consider a real vehicular scenario with more than 600 public buses in the city of Porto [1], where the goal is to predict the number of WiFi sessions per hour in the buses, and therefore, the required bandwidth in the network. The obtained results show that the framework is able to provide the ensemble predictions without any change in the individual predictors, estimating and predicting the behavior of bus users, estimating traveler routes, QoS of the Internet service, traffic conditions, among others [2], thereby improving network management.

978-1-7281-4973-8/20/$31.00

II. Related Concepts

Our proposed framework focuses on the data prediction, allowing the framework to be domain-independent and enabling the data prescription analytics to be done by a domain-specialized client of the framework.

Lately, several forecasting approaches were tested to predict sewer overflow [3]: wavelet neural network [4], and gated recurrent units (GRU) [5]. The LSTM and GRU achieved better results for multi-step-ahead time series prediction, with GRU with a faster learning curve.

At Uber [6], it was developed a forecast model for multi-step time series forecasting that can handle multivariate inputs, to forecast the driver demand for ride sharing on challenging days, such as holidays, where the uncertainty for classical models was high. The new forecasting algorithm showed major improvements (2 to 18% of increase in accuracy) in the forecast when compared with a previous model. The work in [7] presents an architecture that comprises batch processing and real-time analysis. However, it is a general architecture for IoT use cases, and not applied to time series forecasting. Moreover, it is applied independently for each use case. Our proposed approach can be used for different types of time series predictions simultaneously in a straightforward way, without any modification of the API. In [8] it was presented a conceptual general framework that splits the machine learning pipeline in four steps: data collection, preprocessing, learning and interpretation. In our framework, the data collection and interpretation phases are from the responsibility of the service client, preventing the architecture from being an end-to-end framework and turning it into a more flexible and agnostic framework to the data sources and possible interpretations. The most similar work to ours is the one in [9], where the complex implementation of machine learning algorithms is tackled with a service-oriented data-centric architecture, where the machine learning approaches are encapsulated in reusable and extensible microservices. Just like in our work, the configuration of the microservices is done over a REST interface, and there is a training service to improve the models over time. However, there are differences in the architecture and in the implementation. Our framework is focused on forecasting values, and because of that it has a simpler API for forecasting, a dedicated database for real-time values and

other implementation differences. Moreover, we do provide real-time predictions and ensemble approaches.

As far as we know, we could not find in the literature a framework for real-time forecasting. Our framework is modular in its components, since it is implemented with microservices, and it is able to train and use multiple prediction models simultaneously, an essential characteristic for big data architectures when there are many time series systems.

III. Framework Architecture Overview

In a large network, it is required to monitor many different network metrics from various slices in real-time, to be able to improve the network as fast as possible and prevent network problems. Besides real-time forecasting, our framework is also capable of performing multiple forecasts, with a distributed architecture. For a 5G core network with multiple slices, it is of high importance to forecast network metrics separately for each slice, to perform slice-specific actions such as inter-slice resource management or routing modifications. Our proposed architecture allows such division for a large number of slices, according to the physical resources available. In many cases it is more efficient to combine prediction strategies in an ensemble to achieve better results. With the proposed framework, it is also straightforward to create ensemble predictions (predictions with the results of other predictors). The framework includes a message transformer, useful for combining the result of different predictors, or for changing the predictions. The modularity of the framework is adequate for enforcing data privacy policies. The predictor components, where the data is stored, can be relocated to a computational resource with reinforced security.

The high-level view of the prediction framework, depicted in Figure 1, is composed of five main components: the *Prediction API*, the *Predictor Message Broker* (PMB), the *Metric Prediction Component* (MPC), the *Metric Message Transformation Component* (MMTC) and the *Metric Prediction Orchestrator* (MPO).

The *Prediction API* is the interface that allows a client to: (*T1*) predict a value for a time series system/metric; (*T2*) train the predictor for a given metric; (*T3*) save new time series data for later training; and (*T4*) change the training parameters. The Prediction API is a REST microservice that encapsulates in a simple and convenient way the functionalities of the predictors without any operability loss, to be used for non-experts.

All tasks are available as REST endpoints, and they receive the name of the predictor as a parameter. For the task (*T1*), the client should send the current value of the time series to be available for the predictor to use it, along with their timestamp and a list of additional attributes, if needed. The list of additional attributes is optional and depends only on the specific predictor. The API will obtain the result of the predictor and return it to the client, along with the timestamp of the predicted point and any additional attributes. The same input attributes are needed for task (*T3*), but the API saves the point and it does not return data to the client. The parameters that must be sent for the task (*T2*) are the start and end date

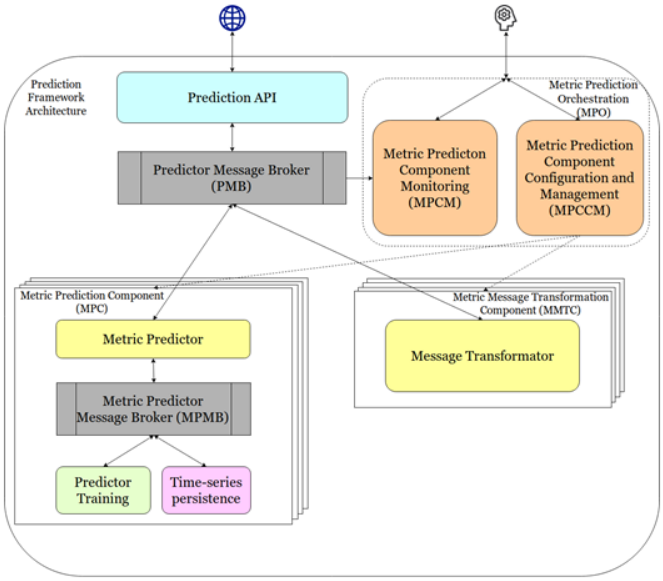

Fig. 1. High-level view of the prediction architecture.

of the data to consider for the training phase, besides any additional attributes. Finally, for the task (*T4*), the parameters are the timestamp of the next training, the period between two training phases, the data period that should be considered for each training and optional additional attributes. Other endpoints are also accessible to rebuild or clear the internal state of the predictor (only applicable when the predictor depends on previous values), and to obtain information about the predictor configurations.

The Predictor API receives the requests from external clients and, according to an internal message protocol, sends a message to the *Predictor Message Broker (PMB)*. PMB forwards the message to the corresponding *metric predictor* or *message transformation component*, enabling the multicast distribution of messages, allowing it to build several prediction components for the same metric, each one with a different prediction model.

The core component of the framework is the *Metric Predictor Component (MPC)*. Each instance of the MPC is responsible for predicting the time series data for a given dynamic system/metric. The MPC is composed by four subcomponents: Metric Predictor (MPC-MP), Metric Predictor Message Broker (MPMB), Predictor Training (MPC-PT), and Time-series Persistence (MPC-TsP).

The MPMB is the internal message broker of the predictor component and it routes messages between the three subcomponents: MPC-MP, MPC-PT and MPC-TsP. The MPC-PT does online or offline training of the model to update it with the newly received data. For offline training, the training task parameters (date of the next training, training data window, etc.) can be adjusted via the Prediction API. The new model will be uploaded to the MPC via MPMB, the internal message broker. The MPC-TsP stores the data to optimize the read and write operations for time-series data, to be used when training

the model.

To enable an ensemble prediction, it is also needed to perform a reduction step, to join the predictor results and perform the adequate transformation. The *Metric Message Transformation Component (MMTC)* performs that transformation by receiving the messages in the prediction message broker and applying custom functions for each predictor component messages, before inserting the transformed messages into the broker. In our use case the transformer will be used to create an ensemble prediction.

The instances of the metric predictor and the transformation components are orchestrated by the *Metric Prediction Orchestration Component (MPO)*. The MPO has two internal components: the Monitoring and the Configuration and Management. The monitoring component can access the logs of the predictor and transformer (the logs are sent to the message broker) and the exchanged messages, to better understand the current state of operation of the various components. The configuration and management module is able to instantiate and manage the life cycle of the predictor and the transformation components. These components can be dynamically added or removed when ordered by the prediction framework manager. The configuration and management component is able to manage the created instances of metric predictors and transformations, ensuring their availability and reliability, restarting them when they crash. When the instances of metric predictors and transformations start their execution, they establish a connection with the message broker (PMB), to be able to report their activity to the monitoring component.

The IoT communication infrastructure in Smart Cities scenarios requires high throughput where failures or high latency may happen. To address these requirements, the implementation of the Predictor API and its components provide fault tolerance, disaster recovery, low response times and high scalability. The proposed architecture has the following characteristics to enable a resilient and distributed framework: a simple and flexible API to be used for clients that do not need to understand the internals of the predictor; easy addition and removal of predictors; creation of ensemble predictions by taking into account the predictions made by multiple models; distributed prediction and training; horizontal scaling; and real-time prediction - the predictions are distributed and performed in real-time, providing a constant prediction of the gathered metrics.

IV. Use cases and Results

The proposed framework is used in a specific 5G network management use case, where the network is divided into slices [10]. The number and the dimension of the slices is variable and depends on the configuration done by the configuration manager and the needs of each slice. We consider three slices: ***S1*** - vehicle-to-infrastructure communication; ***S2*** - control of safety-critical missions (vehicle-to-vehicle communication, remote surgery, public safety platforms, etc.); and ***S3*** - human-machine interaction (virtual reality, high-resolution video streaming, etc).

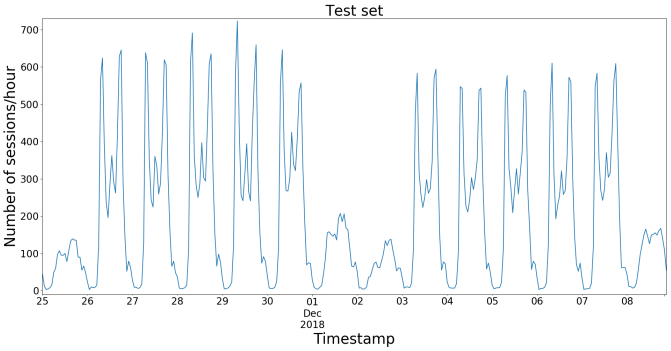

Fig. 2. Number of WiFi sessions per hour in the public buses.

The three slices have different functional requirements: the slice S1 can have a high number of connected devices in a small area (e.g. people connected to the Internet in a transit bus), and the bandwidth of the slice must be adjusted according to the needs of the devices, but without surpassing the needs of other higher priority slices; the slice S2 must have high availability, high reliability and low latency, while most of the times it does not need high bandwidth; and the slice S3 is characterized by real-time data responsiveness and high speed data access. For each slice it is needed a different set of Key Performance Indicators (KPIs) to monitor its behaviour. The use cases show that our predictor framework can be used by the 5G core network orchestration to predict independently the KPIs for each slice.

To test the framework, it is chosen the vehicle-to-infrastructure slice (**S1**) to make predictions. In Porto, in the largest mesh network of connected vehicles, more than 200 000 people enjoy free Wi-Fi every day in buses, taxis and municipal service vehicles [1]. The KPI to predict is the number of sessions for users in the public buses, which corresponds to bandwidth requirements of the slice. The data available is divided in train and test data. The train data is composed of data from 69 days, while the test data has 14 days.

Figure 2 shows that, in the weekend days (25^{th} of November, 1^{st}, 2^{nd} and 8^{th} of December 2018), there is a lower number of sessions than in the week days. Moreover, on the week days there are two peaks per day in the number of sessions, which represent the people's routine of going to work/school and coming from work/school (rush hour).

Different predictors are trained and tested individually in parallel, and the performance metric is the *Root Mean Squared Error* (RMSE) [11]. The three predictor models with lower RMSE are used, which are described below. Other algorithms were trained and tested to model the time-series data, such as Stochastic Gradient Descent, SVM, AdaBoost, and XGB. However, their achieved performance was worse than the three predictor models chosen.

The first predictor uses a *Feed-forward Neural Network*. The inputs of the predictor are the number of sessions in the previous hours. Different hyper-parameters are tested, such as:

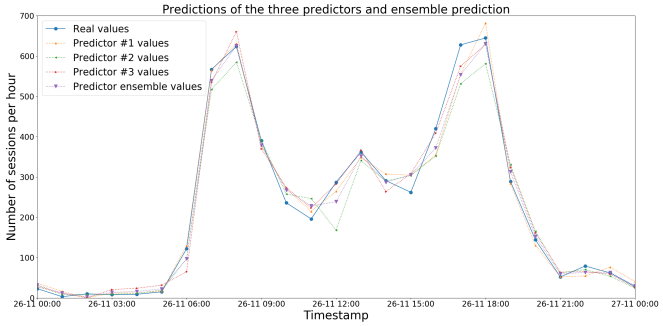

Fig. 3. Number of sessions per hour in 24 hours and the predictions made by the three predictors, as well as the ensemble predictions.

(1) the number of inputs (number of previous hours); (2) the differentiation applied to the input (instead of the raw values of the number of sessions in the previous hours, provide as input the difference on the number of sessions between a specific hour and n hours earlier); (3) or the neural network configuration, which varied from 1 to 4 feed-forward hidden layers with different dropout rates, and from 20 to 200 neurons per layer. Additional features were also tested, such as the day of the week, the hour of the day or a flag to indicate if it is a holiday (holidays have major impact on the number of sessions in the public buses). The best result achieved had 12 hours as input, 1 session as the differentiation value, a neural network with three hidden layers, each one with 200 neurons, a dropout value of 0.2 applied to all hidden layers, and the day of the week as additional feature, with a RMSE of 26.59 sessions per hour. A 24-hour sample of the predictions made in the test set by this Feed-forward Neural Network predictor can be seen in Figure 3, in the *Predictor #1 values*.

The second predictor uses LSTM layers instead of Feed-Forward hidden layers, with the hyperparameter tests being similar. The best achieved result has also 12 hours as input, with no differentiation, with a network with only one hidden LSTM layer with 200 neurons, with no dropout applied to the layers, and a flag to indicate if it is a holiday as additional feature, with a RMSE of 23.4 sessions per hour. A 24-hour sample of this predictor can be seen in Figure 3, in the *Predictor #2 values* (LSTM layers).

Finally, the third predictor has as input 42 features extracted from the time-series, such as the maximum, minimum, median, average or sum of the number of sessions over the previous week of data. In all the tested variations of the algorithm, the Random Forest provided the best results, with a RMSE of 29.19 sessions per hour. A 24-hour sample of this predictor can be seen in Figure 3, in the *Predictor #3 values*.

Additional tests are performed with an ensemble of the previous models. The best results are achieved with the prediction average of the three previous models presented. The ensemble predictions achieved a RMSE of 22.11 sessions per hour. These results are also shown in Figure 3 (*Predictor ensemble values*), with the real values for the number of sessions per hour.

V. Conclusion

This paper proposed a framework for distributed real-time time series forecasting. The framework is characterized by being simple to implement, flexible to the needs of the domain, and scalable to be incorporated in big data architectures. It was implemented and tested in a 5G scenario, with different parallel prediction approaches for the chosen KPIs and different metrics, and considering different slices.

In this paper we discuss results from vehicle-to-infrastructure slice, where the goal was to predict the number of WiFi sessions per hour in the public buses of Porto city. The model that provided the best results involved a combination of three individual predictors, forming an ensemble algorithm. It was shown that the framework was capable of easily providing the ensemble predictions without any change on the individual predictors, improving modularity.

Future work will consider the parallel support of different slices, development of network policies, and the use of the framework for other areas, such as anomaly detection based on the predicted values or characterization of a time series stream.

Acknowledgment

This work is supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project 5G with Nr. 024539 (POCI-01-0247-FEDER-024539)].

References

[1] P. M. Santos *et al.*, "PortoLivingLab: An IoT-based sensing platform for smart cities," *IEEE Internet of Things Journal*, vol. 5, pp. 523–532, apr 2018.

[2] J. Pereira, L. Ricardo, M. Luís, C. Senna, and S. Sargento, "Assessing the reliability of fog computing for smart mobility applications in vanets," *Future Generation Computer Systems*, vol. 94, pp. 317 – 332, 2019.

[3] D. Zhang, G. Lindholm, and H. Ratnaweera, "Use long short-term memory to enhance internet of things for combined sewer overflow monitoring," *Journal of Hydrology*, vol. 556, pp. 409 – 418, 2018.

[4] A. K. Alexandridis and A. D. Zapranis, "Wavelet neural networks: A practical guide," *Neural Networks*, vol. 42, pp. 1 – 27, 2013.

[5] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.

[6] N. Laptev, J. Yosinski, E. L. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *Proceedings of the Thirty-fourth International Conference on Machine Learning*, ICML'17, 2017.

[7] P. Ta-Shma *et al.*, "An ingestion and analytics architecture for IoT applied to smart city use cases," *IEEE Internet of Things Journal*, vol. 5, pp. 765–774, apr 2018.

[8] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, "Machine learning for smart building applications: Review and taxonomy," *ACM Computing Surveys*, vol. 52, 02 2019.

[9] M.-O. Pahl and M. Loipfinger, "Machine learning as a reusable microservice," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Apr. 2018.

[10] 3GPP, "System architecture for the 5G system (5GS)," 2017.

[11] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679 – 688, 2006.