Towards an Ensemble of Clusterers to Improve Forecasting for Self-Driving Data Management

S M Andalib Hossain
Otto-von-Guericke-Universit Magdeburg
Magdeburg, Germany
s.hossain@st.ovgu.de

Kurian John
Otto-von-Guericke-Universit Magdeburg
Magdeburg, Germany
kurian.john@st.ovgu.de

Ahmad Abdullah Tariq
Otto-von-Guericke-Universit Magdeburg
Magdeburg, Germany
ahmad.tariq@st.ovgu.de

Le Anh Trang
Otto-von-Guericke-Universit Magdeburg
Magdeburg, Germany
anh1.le@st.ovgu.de

ABSTRACT

Being popularly used previously in the context of finance and business, *time series forecasting*, however, has been nowadays applied in various domains. One of them is for self-managing database systems. A recently proposed approach in this field, which consists of clustering queries and then forecasting the cluster centers, is discussed in our paper. We want to be able to allocate resources to a user in the future depending upon their usage in the past. The problem is that data for forecasting is too much (there is a large amount of fine-grained queries, and abundant information about their resource usage), and hardly all is relevant. In this paper, we will discuss two clustering techniques we used to reduce data. One technique is based on the literature for the use case, another is tried out for the first time by us. We report that several aspects of clustering can have an impact on the accuracy of the resulting forecasts. Based on our observations we propose as future work to evaluate a novel way to potentially improve forecasting for self-driving database systems, by keeping an ensemble of clusters.

KEYWORDS

Time series clustering, Time series forecasting, Time series for data management, LSTM, DTW, DBSCAN

ACM Reference Format:
S M Andalib Hossain, Kurian John, Ahmad Abdullah Tariq, and Le Anh Trang. 1997. Towards an Ensemble of Clusterers to Improve Forecasting for Self-Driving Data Management. In *Proceedings of (Modern Database Technologies).* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

After the web 2.0 the Internet usage increased significantly. With more and more people using Internet services everyday, it has become essential to optimize resource allocation for users requesting data from servers.

Modern Database Technologies, July 2018, Magdeburg, Germany

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

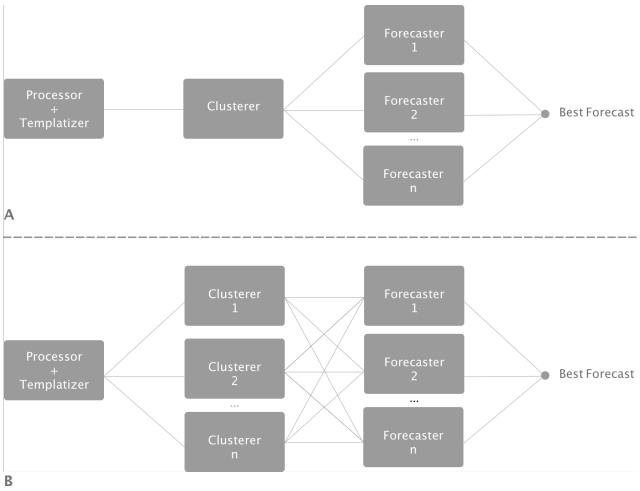

Figure 1: A represents the approach proposed in QueryBot5000, whereas B represents the one introduced by us.

At first, scientists relied on hardware solutions, using a larger number of processors and more powerful processing. But with the web 3.0[1], the technological advances opened a new front. With so much data already accumulated on servers; including performance data, we can use this data to analyze and understand how people use the Internet and, generally, shared computer systems. Providing servers with solutions to optimize resources, avoiding over or under provisioning. One such solution can be done by predicting the exact amount of resources that will be needed in any specific moment, based on past usage data[17].

For the purpose of these optimizations, a special kind of forecasting called *Time Series Forecasting* has been employed, where the usage data is modeled as a time series. But before introducing time series, lets take a step back and understand the concept of forecasting first. Forecasting is the process of making predictions about the future, using previous data; it has typically been used in the financial domain [4, 8]. A typical time series has values of

[1] Web 3.0 or Semantic Web says data derives the web today. This standard allows the frameworks to share and reuse data, including performance data, and this data can be processed by machines.

any entity on one dimension and time on the other[2], therefore forecasting is about projecting the values of an observed time series into the future, by using specialized models.

In this project we are going to focus on forecasting time series of the amount of queries of a given type that arrive to databases. This number of queries is also the workload of the database. Since databases can receive a lot of requests, we need some techniques to reduce the size of data we are going to process for forecasting. In order to reduce the amount of data we are going to use *clustering*. This is an approach proposed in closely related work [11]. Given a large amount of data, a clustering algorithm will produce data points that represent all major classes of the data. These points are usually called cluster centers or prototypes. By forecasting on the cluster center, instead of all the points (time series for queries) that belong to the cluster, we are able to reduce the amount of operations, making the process suitable for a live system.

As we have mentioned, a similar approach has already been developed for forecasting time series for data management in QueryBot5000 (or QueryBot5000) [11]. As the Figure 1 shows, in our research we consider the basic research question: whether there is an impact of using different clustering approaches on the accuracy of the resulting forecasts? Should there be an impact we would like to establish if the current approach of having a fixed clustering followed by forecasting [11] could be improved by an ensemble of clusterers (Figure 1).

In the paper we document a series of experiments that we did on real-world traces[3] from databases, reducing the amount of time series by using different clustering methods; and next evaluating the forecasts using an *LSTM* forecaster developed by us and the solutions provided by QueryBot5000[11].

Our contributions can be listed as follows:

- Based on two datasets, including real-world database traces, we evaluate the goodness of forecasts based on clustering with different methods that we implement after preparing the datasets. We can report that clustering approaches can make a difference in the forecasts, which suggests that there is room for improvement over the proposal of QueryBot5000[11] by considering alternative approaches for forecasting.
- Second we evaluate a single clustering technique, but with different similarity functions. For this we also change the clustering code of QueryBot5000[11], such that their incremental DBSCAN algorithm forms different cluster centers according to the functions. We find that Dynamic Time Warping overall leads to improvements over basic cosine similarity for this algorithm. Through this we can confirm that there are also possible improvements in QueryBot5000[11] through this change.
- Still we observe that there is room for improvement since different clustering configurations benefit time series differently. Thus we can propose a novel approach that we call *ensemble clustering*, which can constitute an improvement over state-of-the-art forecasting solutions for self-driving DBMSs, helping to find better forecasts by keeping alternative clusters available. However our solution needs to be developed beyond the scope of a student project.

We structure our paper in the following way: section 2 describes time series, its components and an overview of time series clustering. Then we discuss the process of forecasting DBMS workloads in section 3. Section 4 presents our experiments performing clustering and forecasting techniques and finally section 5 summarizing the observations and future directions in the conclusion.

2 BACKGROUND

A time series is defined as a sequence of data points observed over a consecutive period of time [1]. It consists of three components, as shown in Figure 2: trend, seasonality and noise (uncaptured remainder). Trend is the general tendency of a time series to increase, decrease or stagnate over a long period of time. Seasonality represents the periodic fluctuations within a time (day, month,year) during the season. And the noise corresponds to unpredictable influences, which are not regular and also do not repeat in a particular pattern.

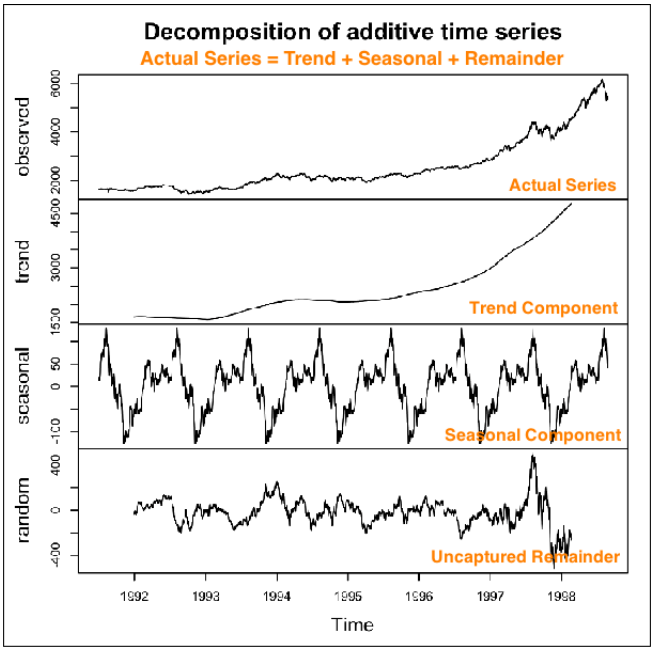

Figure 2: Components Of Time Series. Source http://rstatistics.net/time-series-analysis

With huge data sets we have to process data fast to predict the resource usage of a server. If we process all the data and cannot predict resource usage in a timely manner, then the prediction is of no use. Therefore we need to reduce the data set, so that we do not lose important instances and yet are able to predict accurately over near real-time data. For that purpose we will use different clustering techniques.

Clustering is a *Machine Learning technique*, in which data points with similar properties are grouped together.

[2]Uni-variate time series have only one dimension next to time, Multi-variate have several.

[3]Traces are the collection of all accesses to database items by queries and transactions, during an observation period.

Using the reduced data set we have a task in hand to forecast. For this purpose we are going to see different algorithms later on, but first we discuss on clustering.

Aghabozorgi et al. [2] give an overview on the state of the art in time-series clustering. According to their observations, time-series clustering can be first classified into three approaches: the first one considers the whole time series, which is the case in our paper, another based on subsequences of time-series [10] and the last one is about time point clustering (where each time point of a time series is clustered into a group, which is similar to segmentation and can aid in understanding the series). There exists a criticism by Keogh and Lin [10] that clustering on time series subsequences is meaningless; for this, authors propose a novel method considering time series motifs, which are groups of individual time series or subsequences that have similar shapes. In our work and in QueryBot5000[11], the focus is on clustering for whole time series.

Abjabozorgi et al. [2] provide examples of clustering being applied in different domains, and discuss the pros and cons of each proposed solution from the literature.

Authors describe the existing techniques in several dimensions: the chosen representation methods, similarity/distance measurements, clusters prototyping (i.e. how the cluster centers are formed based on the series belonging to the cluster), clustering algorithms as well as evaluation methods. Essentially each clustering process can start by approximating data using a representation method, the clustering algorithm can proceed next by employing a similarity function and a prototype calculation step (where a cluster center is formed, helping to summarize the cluster). To conclude the resulting clustering can be evaluated according to different criteria.

Representation methods seek to reduce the data required to represent a single time series. This is achieved through transformations, like the Discrete Fourier Transform, the Discrete Wavelet Transform, and others. In both our implementation and QueryBot5000[11], we use the raw time series data and we do not try transformations, which could also improve the process. This is left as future work.

Regarding similarity measurements, various methods are mentioned and among them, Dynamic Time Warping (DTW), which has been discussed later in our paper, is asserted to be highly efficient in finding time series with similar patterns. QueryBot5000[11] uses cosine similarity.

Through the paper of Abjabozorgi et al. [2], it can be seen that time series clustering has received much attention with dozens of researches and published papers over the last 20 years. Authors discuss various time-series clustering algorithms as well as their advantages and disadvantages in specific cases. In our project, two of those approaches are adopted comprising: partitioning (k-Means) and density-based (DBScan).

Cluster prototyping is another important choice. There are at least three ways of finding a clusters representative: the usage of medoid sequence, the local search prototype, and the average sequence. Our work employs the third approach of utilizing averaging time-series as cluster prototype with Dynamic Time Wrapping [4], but we additionally take into consideration the Barycenter-Averaging method (in K-Mean DBA) to help improving the cluster's center.

[4]QueryBot5000 does the same for the time series in a cluster.

Abjabozorgi et al. [2] suggest that some valuable criteria from clustering algorithms are the following: the ability to be incremental (making it easy to add new data), robustness to noise, support for multi-variate series, use of data in a summarized representation (not raw data, which authors consider to be inefficient in the use of memory), support for unequal time series (with different lengths), support for anomaly detection and use of important points for the clustering.

In this section we provided a brief background on clustering of time series and gave reference to sources where more information can be found. In the next section we describe, step by step the workflow for workload forecasting proposed for QueryBot5000 [11], and we highlight our alternative implementations for some components.

3 WORK-FLOW FOR FORECASTING DBMS WORKLOAD

3.1 Preprocessor

In our project we are using the same preprocessing component of QueryBot5000 [11], where queries of identical templates (i.e, which are formed after removing some tokens and individual information from queries) are counted to predict the workload. There is a notable reduction of data in each template because there only remains arrival rate information of queries and basic descriptive features, but no individual query. Here, collecting and combining steps of our project, same as for QueryBot5000 is described:

- Step 1: Replace all constants of queries by a placeholder "$".
- Step 2: Normalize all the queries by using same spacing, case, and bracket placement.

The outcome are generic query templates. They contain the total count of the generic query and their timestamps.

3.2 Clusterer

Though the pre-processing techniques decrease the total queries numbers, templates are not easily usable to build a model because using each template still requires an important amount of time. To fasten the training period templates are clustered according to their temporal features (if they arrive in a similar rate), instead of their logical features. In fact Ma et al. [11] show that temporal features provide better forecasts than logical features.

The clustering technique adopted by QueryBot5000 [11] is a variation of DBSCAN supported by a Kd-Tree. DBSCAN [6] is a clustering algorithm based on the density of points. For each member of a cluster, the neighborhood of a given radius *Eps* has to contain at least a minimum number of points *MinPts*. A set of density-connected objects are the members of a cluster and not connected members are considered as noise.

The variation with original DBSCAN is that in DBSCAN a cluster member is decided by measuring its distance with any core point of the cluster whereas, in QueryBot5000, a point is assigned to a cluster based on distance measure with the cluster center.

To put the templates in the same cluster, an arrival rate similarity threshold ($0 \leq \rho \leq 1$) was used. Three steps of incremental clustering technique of QueryBot5000[11] are discussed below:

- Step 1: If a template has similarity score with any cluster center that is greater than threshold ρ, it will be assigned to that cluster. If there are more than one cluster, with which similarity score is greater than threshold ρ, the template is assigned to cluster with the highest similarity. Kd-tree is incorporated in order to speed up the process of finding the closest cluster. The center of a cluster is then updated by averaging all of its templates. In case there exists no cluster that is close enough to the coming template, a new cluster is created for it.
- Step 2: In this step the similarity between the new cluster center and all previous templates of the cluster is checked. If the similarity of any template is less than ρ, the template is removed from the cluster and the previous step (1) is repeated to find out the cluster for that template.
- Step 3: The similarity among all the cluster centers is measured. If there is a similarity higher than ρ, two clusters are merged.

Based on QueryBot5000[11] clustering technique, in order to evaluate if the similarity function has an impact on performance, we utilize *Dynamic Time Warping (DTW)* instead of cosine similarity as the distance function.
The concept of Dynamic Time Wrapping using in sequence comparison is probably the most well-known similarity measurement nowadays for time series. It is considered to substantially outweigh other distance functions such as: Euclidean, Manhattan, etc because of its ability to match the the time series with the same pattern but in different phases regarding the time domain. The core idea is that time is warped such that if there are points that match close they are shifted to be closer (shifting in turn the complete sequence), and the smallest distance is calculated among those closer points. Figure 3 gives an example of this approach.

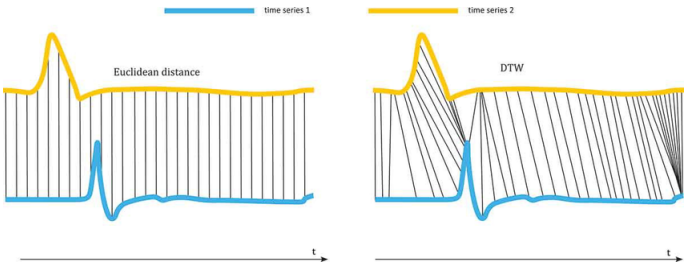

Figure 3: Euclidean distance and Dynamic Time Wrapping alignment. Source: [15]

In this algorithm, a warping matrix is constructed between each pair of time series and their Dynamic Time Wrapping distance is calculated as the sum of all cells traversed by the global optimal path as in the Figure 4

We also introduce another clustering technique that makes use of Dynamic Time Wrapping and includes additionally the superior Barycenter-Averaging technique for improving prototyping shape-based time series : K-Means DBA
The original K-Means is known as a simple and high-speed clustering technique, which includes two main steps:

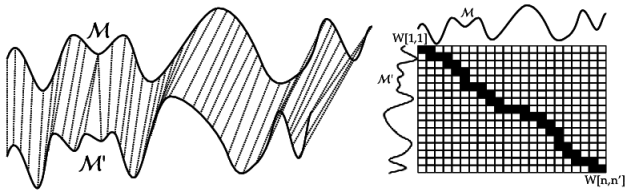

Figure 4: Warping matrix and optimal alignment path. Source: [3]

- Step 1: Initialize randomly K numbers of centroids
- Step 2: Assign data points to the closest centroid (based on Dynamic Time Wrapping distance) and update the centroids using Barycenter Averaging

These tasks are repeated until there are no changes in the positions of K centroids.
The concept of Barycenter-Averaging is proposed by Petitjean et al. [14] and intends to refine average sequence of a cluster, or in other words, turn the initial average sequence into a Barycenter, such that sum of squared Dynamic Time Wrapping distances from it to all the time series within the cluster is minimum. Each iteration is a refinement consisting of two steps:

- Step 1: Compute Dynamic Time Wrapping between the initial average sequence and the set of considered time series in order to find the temporal alignments among them. Do this with every time series within the cluster.
- Step 2: This alignment is then updated by adjusting every index of the average time series, so that it becomes a barycenter of all the indexes correlated to it during the previous step.

The number of iterations is defined by user. Figure 5 shows an example of Barycenter-Averaging with 4 iterations. As can been seen from this figure, there are associations between every index of two given time series and the average one (time series in the middle). After each iteration, each index of the average sequence becomes barycenter of the associated indexes of the given set of sequences. The sum of squared, accordingly, is minimized locally and leads to minimum total sum of squared Dynamic Time Wrapping distances. The returned barycenter after four iterations is showned to be considerably refined.

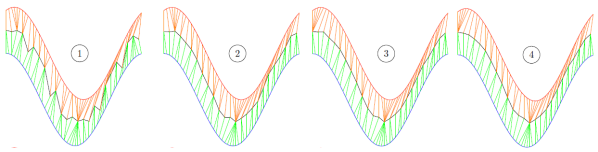

Figure 5: Dynamic Barycenter Averaging iteratively adjusting the average of two sequences. Source: [14]

3.3 Forecaster

In QueryBot5000 [11], a combination of Linear Regression (LR) and Recurrent Neural Network (RNN) is used for forecasting named an *ENSEMBLE model*. In LR, to get global optimum, linear models do not have to perform an extra optimization step. On the other

hand, RNN is effective to predict non-linear systems. So, Ma et al. [11] proposed an averaging method of both of them. They also tried weighted averaging method but that caused over-fitting and more frustrating results. Moreover, it is not fruitful to forecast the periodic spikes in the workload that are far apart from each occurrence.

In our paper, we focused on only a special kind of RNN: long short-term memory(LSTM) [9] that can remind values from previous steps for use in future. It is also useful for predicting nonlinear data where Linear Regression fails dramatically.

4 EXPERIMENT

In the following section, we present our experiment results as our answer to the research questions formulated. For forecasting, apart from QueryBot5000 [11] , we use an LSTM network consisting of 4 layers, the first of which is the LSTM layer consisting of 4 LSTM neurons with 1 look back, and 3 extra layers of 1 neuron each which is useful for scaling. The model is trained for a small number of 50 epochs on batches of 2 samples. We divide our data into 67% training and 33% testing. For the first research question we used a *Web Traffic Time Series Forecasting* data-set from Kaggle for clustering and forecasting, and for the second research question we used the *Bus Tracker* database trace shared by the developers of QueryBot5000[11].

4.1 Impact of Changing Clustering Technique on Forecasting Accuracy

So our motivation for this experiment was to see, *how big is the influence of changing clustering techniques (instead of the features) on the forecasting accuracy?* In the QueryBot5000 [11] authors only discuss one clustering approach. That is a variant of DBSCAN, with cosine similarity as the function for reducing the time series for forecasting.

So, in our experiment we used another clustering technique: K-Means with Dynamic Time Wrapping, for clustering the time series to test whether it makes any difference in the forecasting accuracy. For this firstly, we fed the *Web Traffic Time Series Forecasting* data-set from Kaggle into the QueryBot5000[11] clustering method and the result of the clustering algorithm was forecasted using our LSTM forecasting model.

Secondly, the same dataset was clustered using a K-Means algorithm that uses Dynamic Time Wrapping as the similarity measure and the resulting cluster centers were forecasted using the same LSTM forecasting model.

To compare the accuracy of both forecasts, we measured the *Mean Absolute Percentage Error (MAPE).* As equation 1 describe, where t is the time and p_t is the percentage error at time t. Where Figure 6 present the visualization of the experimental results for MAPE. These show that K-means is able to achieve lower overall errors than the approach followed by QueryBot5000[11] for this dataset.

$$MAPE = mean(|p_t|) \tag{1}$$

Since the values are calculated as the output of different clustering techniques, for comparison we plotted them in the descending order. From Figure 6, it can be seen that some clusters of the

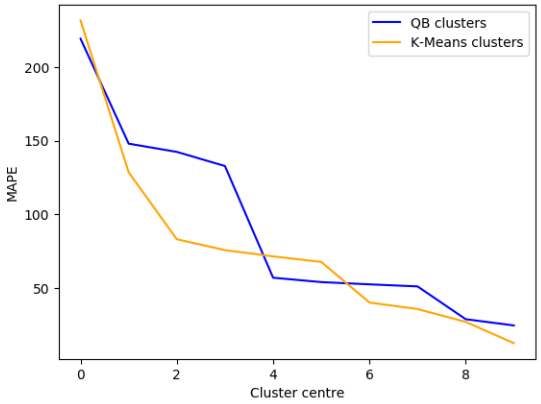

Figure 6: Mean Absolute Percentage Error

K-Means algorithm perform better than the QueryBot5000[11] clusters. Figure 7 shows the best and worst clusters formed by QueryBot5000[11] clustering technique and Figure 8 shows the best and worst clusters formed by K-Means clustering technique using Dynamic Time Wrapping as similarity measure. Thus it can be seen that a different clustering algorithm with a different similarity function might result in better forecasting overall. Thus, our experiment justifies the methodological approach of studying different clustering techniques in addition to different forecasting approaches, when there is the quest to find the better forecasts for a set of time series.

4.2 Impact of Different Similarity Function

Our second question was to analyze:*can the state-of-the-art clustering approach of QueryBot5000[11] be improved through changing the similarity function?* Since a different clustering approach had an effect on forecasting, we thought that maybe different similarity measure might also contribute in better forecasting accuracy. So in QueryBot5000[11] we changed the similarity function to *Dynamic Time Wrapping* to test whether it performs better then the existing method. For the experiment, we used the sample database traces provided by the developers of QueryBot5000[11], they consisted of data using a bus-tracking app.

The sample database traces were clustered using the original QueryBot5000[11] clustering technique and also using the modified QueryBot5000[11] clustering technique, to use our similarity measure. The resultant cluster centers were forecasted using the ensemble forecasting model of QueryBot5000[11] and also using our implementation of the LSTM forecasting model.

Figure 9 and Figure 10 show the forecasted results of cluster centers calculated by the ensemble forecasting model of query bot over the centers for the two clustering techniques. The images correspond to the top 3 cluster centers, which are 1000 hours (with each tick equal to one hour) from the 12 December till the 24th of January (this corresponds to 41.6 days). The forecasted values of both QueryBot5000[11] cluster centres and the cluster centres of the modified version of QueryBot5000[11] are generally able to

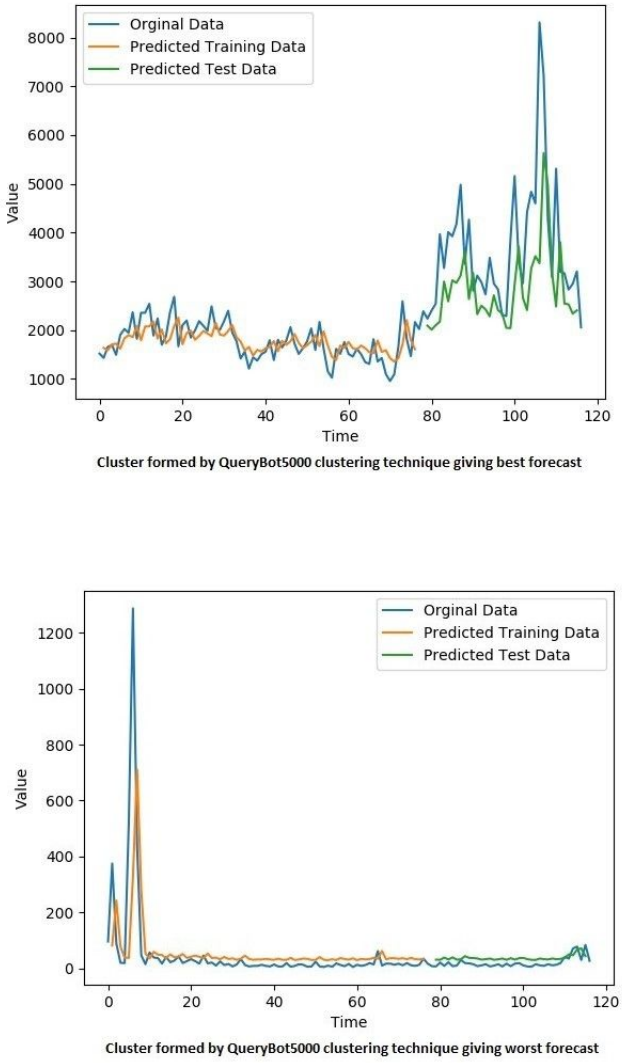

Figure 7: Best and worst forecasts of QueryBot5000

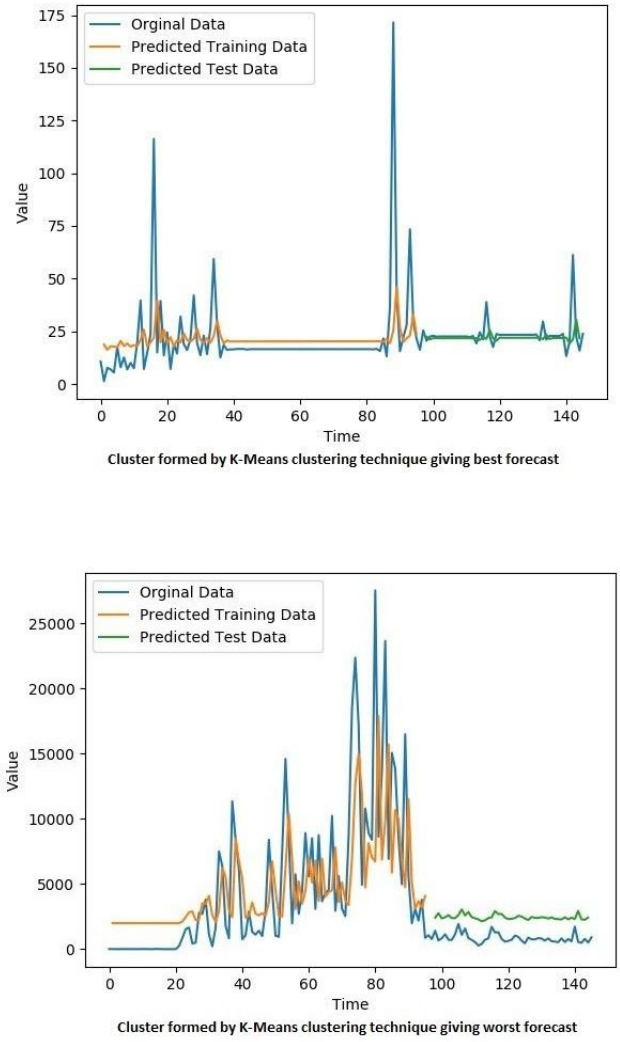

Figure 8: Best and worst forecasts of K-Means

follow the time series trends but there are two issues, first a large spike and second, some problems in the prediction in the second half of the time presented.

There is a large spike in the forecast occurring around the 31st of December. This does not correspond to the reality of that date, where the requests are very less. For such cases hard to predict occurences), the authors suggest that Kernel Regression would work well.

On the January (the last observations), there is a mix of over and under-prediction suggesting that the model does not work too well.

Figure 11 and Figure 12 show the results of the LSTM forecasting model for two clustering algorithms. The data used here corresponds to 55 days from the 29th of November until the 23rd of January, with a tick grain of 10 minutes. The forecasted values of both QueryBot5000[11] cluster centres and the cluster centres of the modified version of QueryBot5000[11] are able to follow the trend of the original time series but both the forecasts are not able to predict the arrival rates in the upper ranges of the series. This is specially notable when the ranges for the time series are small, and when there are spikes. For the latter we have already mentioned that the authors recommend the use of Kernel Regression over LSTM.

The use of the modified version leads first, to a selection of cluster centres where spikes are not observed, making the centres amiable for forecasting with LSTM or techniques other than those specific to spikes. The resulting centres can also be forecasted better, leading to less errors. Hence our second result shows that on real

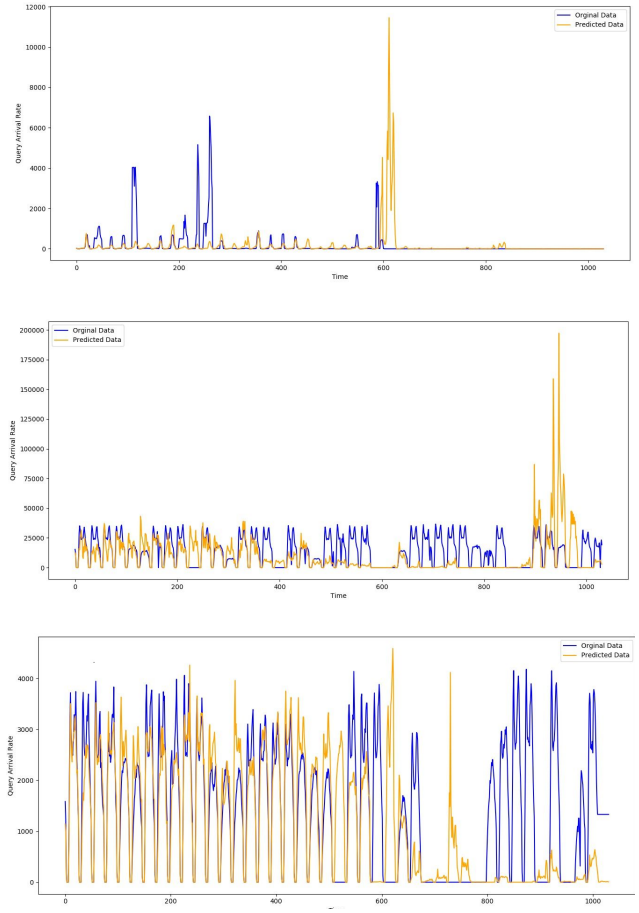

Figure 9: The forecasted result of cluster centers calculated by the modified (using Dynamic Time Wrapping) QueryBot5000 clustering algorithm by an ensemble forecasting model

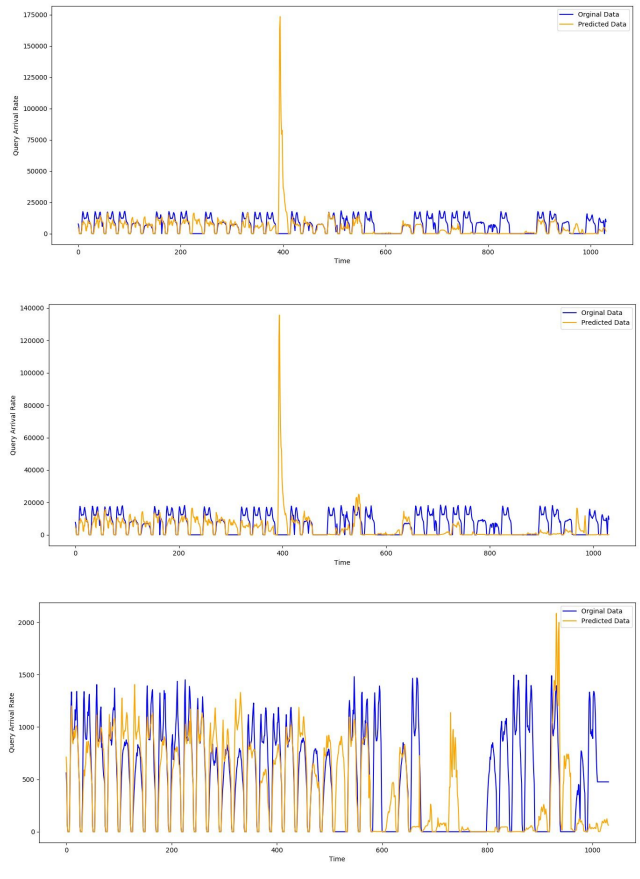

Figure 10: The forecasted result of cluster centers calculated by the QueryBot5000 clustering algorithm by ensemble forecasting model

database traces changing the similarity function of the clustering algorithm can lead to different cluster centers, some of which can include patterns with less spikes, as seen in the LSTM results.

These are, of course, only early observations and more tests are needed to test if Dynamic Time Wrapping will always lead to better cluster centres. Nonetheless, our results show that there are promising improvements by considering alternative cluster centres. Hence these results also sustain the proposition of an ensemble of clusterers supporting forecasting.

4.3 Related Work

To our knowledge time series have not been traditionally used for modeling workloads and aiding data management, instead static models (i.e., not considering temporal aspects) or models for sequences (Markov models) have been frequently used in the literature.

The survey of Calzarossa et al. helps in understanding the state of affairs on workload characterization for databases (i.e., the construction of a model for the workload, based on observations) circa 1993 [5]. Authors discuss the existence of work employing finite state machines, graphical displays and simple numerical summaries, to portray patterns in the data that can be used as input to characterize a workload. Clustering of queries or jobs was performed statically, based on their features, without consideration of their arrival times. Authors also discuss work that proposes Poisson models to approximate a time series of database events. Zhang et al. report a more recent state of affairs [20]. In their paper a clear separation of tasks is provided, between workload characterization (static vs dynamic) and some ways, in which this can be used (scheduling, admission control, execution control). Authors report, based on product documentation, about how these tasks are performed in different commercial systems, like IBM DB2 Workload Manager, Microsoft SQL Server Resource/Query Governor and Teradata's Active System Management. These systems rely mostly on static (i.e., dynamics unaware) classification methods (to assign a query to a labeled workload group); in some tools the classification is expected to take place offline, with users providing the database with the parameters for a classifier model. Tran et al. [18] in the context of Oracle's Workload Intelligence, consider the use of Markov

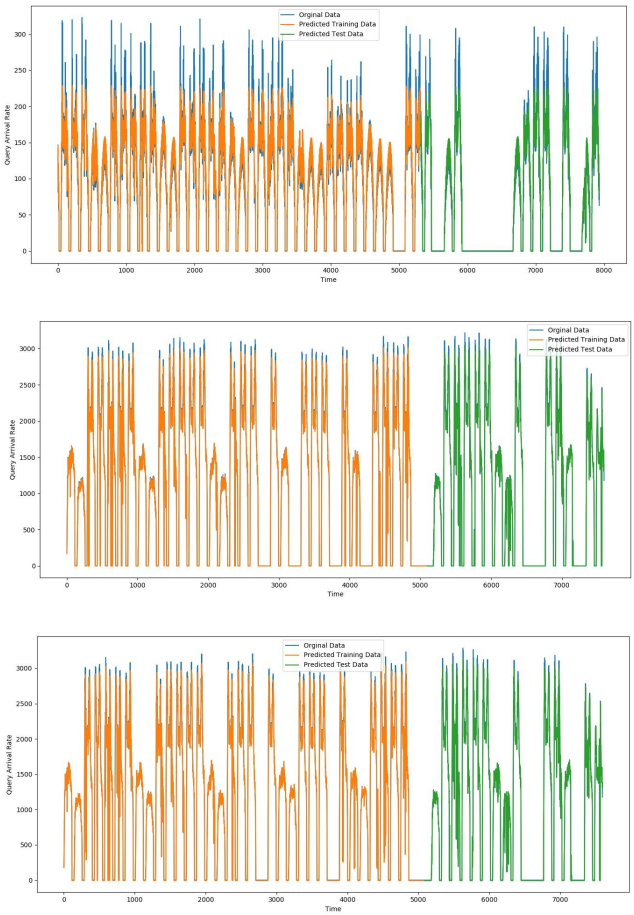

Figure 11: The forecasted result of cluster centres calculated by the modified (using Dynamic Time Wrapping) QueryBot5000 clustering algorithm by LSTM forecasting model

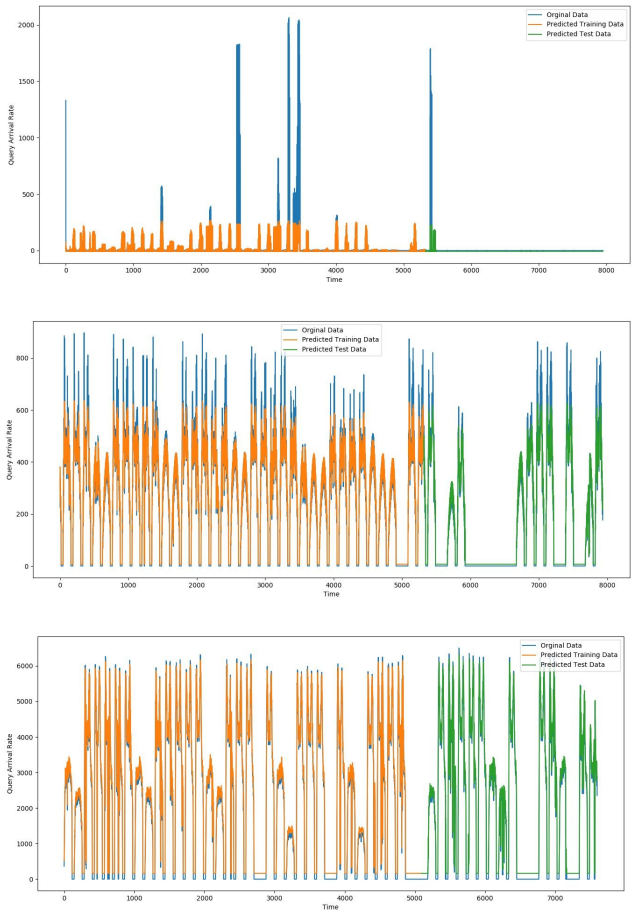

Figure 12: The forecasted result of cluster centres calculated by the QueryBot5000 clustering algorithm by LSTM forecasting model

models to represent a workload compactly, modeling that actions are likely to occur given an observed action. These models are built by mining sequences from database traces. Authors propose that such a solution can aid in predicting patterns likely to be observed, shifts in workloads and in generating artificially realistic workloads. Holze et al. [7] report a similar study where Markov models are used to detect shifts in workloads, such that administrators can be alerted on the need for making some changes to the physical design or other configurable knobs.

In DBSeer Mozafari et al. [12] develop techniques for statically classifying workloads and, additionally, predicting aspects such as the time expected for picking a lock in an environment with contention.

Pavlo et. al [13] also report more work using static and Markov models.

In recent years there has been a growing amount of work that uses time series, rather than Markov or static models, to represent workloads and analyze them. Such work is properly referenced in the papers of Pavlo et al. [13] and Ma et al.[11].

Causality has been studied with time series, aiming to help database administrators in performing root-cause analysis. DBScherlock [19] - is a performance explanation framework from the University of Michigan that builds upon the work of the same authors in DBSeer. They use time series data extracted from the query logs for the purpose of diagnosis and explanation of performance problems by building causal models in an online transaction processing (OLTP) database. The potential causes of these issues are then presented to the database administrators for root-cause identification, which is afterwards re-integrated into the system to refine the future diagnosis.

Forecasting for managing cloud scaling of data systems has also been studied with time series. Roy et al. propose a system that employs the ARMA model [16]. P-store is a recent system that integrates predictions into a scalable cloud database [17]. Authors employ an auto-regression model, showing that better compute usage and latencies can be achieved by predictive scaling. Pavlo et al. [13] also discusses work that forecasts for cloud systems.

QueryBot5000 [11] is a robust forecasting framework that enables the database management system to foresee the expected load of queries coming in the future. In general, raw SQL queries and their arrival rates are processed by a combination of techniques from pre-processing, templatizing to clustering, which are later forwarded to the Forecaster to predict the arrival rates of upcoming queries. QueryBot5000[11] also stands out among existing work by forecasting on different horizons, thus working potentially well for different kinds of optimizations. Our work is based on this tool, and we employ the same traces, albeit only a subset of them, for our experiments. In our study we show that the forecasting formulation proposed by the authors can indeed produce accurate forecasts. Building on this we evaluate if the approach of the authors can be still improved by changes in the clustering; and more specifically by keeping an ensemble of clusterers. Hence we open an area of research on how to support simultaneous clustering for forecasting in DBMSs.

5 CONCLUSION AND FUTURE WORK

Aghabozorgi et al. [2] categorized the building blocks of time series clustering to involve representation, similarity functions and how the cluster center is calculated. The combination of choices for these blocks determines how a clustering algorithm proceeds. In this paper we apply these insights in studying a recently proposed workflow [11] for developing a self-driving database component through the use of forecasting, concretely through the steps of pre-processing, templatizing, clustering (to reduce the number of time series to forecast) and finally forecasting by using an ensemble of methods.

In our work we specifically focus on the clustering component, and we consider whether there is an impact or not, on the resulting quality of the forecasts, of changing the core components for clustering. We find that indeed clustering algorithms, can lead to different cluster centers and hence to different goodness in the resulting forecasts.

In addition, we find that while keeping a same clustering technique, but changing the underlying similarity function, we can reach different results.

Our findings hold practical impact for how forecasting should be developed for self-driving database components. We envision that an ensemble of clustering techniques could be used, next to the ensemble of forecasters, to reach better forecasts. In future work our idea needs to be evaluated by adopting specific actions that can employ our forecasts; also by evaluating alternative representations for the clustering. We can also improve our work by studying the impact when using different horizons, and by considering how to optimize the runtime performance of the calculations such that they could be used in a live self-driving DBMS. Our expectation is that our current work can open a research direction to improve the state of the art in forecasting, as exemplified by Ma. et al [11].

The physicist Niels Bohr is attributed to saying that it is difficult to make predictions, especially when they are about the future. In this paper we hope to have shown that predictions about the future might not be that difficult after all if we start from an ensemble of clusterers.

ACKNOWLEDGMENTS

We would like to specially thank our supervisor, Gabriel Campero Durand, for his great support, encouragement and precious advice he has given us throughout our whole project. Besides, we would also like to express our sincere thanks to Lin Ma and the research team of Andy Pavlo for providing us a sample of their trace datasets as well as useful guidance for working with QueryBot5000.

REFERENCES

[1] Ratnadip Adhikari and R. K. Agrawal. 2013. An Introductory Study on Time Series Modeling and Forecasting. *CoRR* abs/1302.6613 (2013). arXiv:1302.6613 http://arxiv.org/abs/1302.6613

[2] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–A decade review. *Information Systems* 53 (2015), 16–38.

[3] Michal Balazia, Jan Sedmidubsky, and Pavel Zezula. 2014. Semantically Consistent Human Motion Segmentation. In *Database and Expert Systems Applications*, Hendrik Decker, Lenka Lhotská, Sebastian Link, Marcus Spies, and Roland R. Wagner (Eds.). Springer International Publishing, Cham, 423–437.

[4] Danko Brezak, Tomislav Bacek, Dubravko Majetic, Josip Kasac, and Branko Novakovic. 2012. A comparison of feed-forward and recurrent neural networks in time series forecasting. In *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012 IEEE Conference on*. IEEE, 1–6.

[5] Maria Calzarossa and Giuseppe Serazzi. 1993. Workload characterization: A survey. *Proc. IEEE* 81, 8 (1993), 1136–1150.

[6] Dimitris Plexousakis Vassilis Christophides Manolis Koubarakis Klemens BÃűhm Elena Ferrar Elisa Bertino, Stavros Christodoulakis. 2004. Advances in Database Technology - EDBT 2004. In *9th International Conference on Extending Database Technology, March 14-18, 2004*. Springer Berlin Heidelberg, Heraklion,Crete, Greece, 94.

[7] Marc Holze and Norbert Ritter. 2007. Towards workload shift detection and prediction for autonomic databases. In *Proceedings of the ACM first Ph. D. workshop in CIKM*. ACM, 109–116.

[8] Chao Huang, Li-li Huang, and Ting-ting Han. 2012. Financial time series forecasting based on wavelet kernel support vector machine. In *Natural Computation (ICNC), 2012 Eighth International Conference on*. IEEE, 79–83.

[9] Patterson J. 2017. In *Deep Learning: A PractitionerâĂŹs Approach*. OâĂŹReilly Media.

[10] Eamonn Keogh and Jessica Lin. 2005. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and Information Systems* 8, 2 (01 Aug 2005), 154–177. https://doi.org/10.1007/s10115-004-0172-7

[11] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J Gordon. 2018. Query-based Workload Forecasting for Self-Driving Database Management Systems. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 631–645.

[12] Barzan Mozafari, Carlo Curino, and Samuel Madden. 2013. DBSeer: Resource and Performance Prediction for Building a Next Generation Database Cloud.. In *CIDR*.

[13] Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C Mowry, Matthew Perron, Ian Quah, et al. 2017. Self-Driving Database Management Systems.. In *CIDR*.

[14] François Petitjean, Alain Ketterlin, and Pierre Gançarski. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition* 44, 3 (2011), 678–693.

[15] Peter Ranacher and Katerina Tzavella. 2014. How to compare movement? A review of physical movement similarity measures in geographic information science and beyond. *Cartography and geographic information science* 41, 3 (2014), 286–307.

[16] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. 2011. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 500–507.

[17] Rebecca Taft, Nosayba El-Sayed, Marco Serafini, Yu Lu, Ashraf Aboulnaga, Michael Stonebraker, Ricardo Mayerhofer, and Francisco Andrade. 2018. P-Store: An Elastic Database System with Predictive Provisioning. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 205–219.

[18] Quoc Trung Tran, Konstantinos Morfonios, and Neoklis Polyzotis. 2015. Oracle workload intelligence. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1669–1681.

[19] Dong Young Yoon, Ning Niu, and Barzan Mozafari. 2016. DBSherlock: A Performance Diagnostic Tool for Transactional Databases. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. ACM, New York, NY, USA, 1599–1614. https://doi.org/10.1145/2882903.2915218

[20] Mingyi Zhang, Patrick Martin, Wendy Powley, and Jianjun Chen. 2018. Workload Management in Database Management Systems: A Taxonomy. *IEEE Transactions*

on Knowledge and Data Engineering 30, 7 (2018), 1386–1402.