Predicting Complex Events for Pro-Active IoT Applications

Adnan Akbar, Francois Carrez, and Klaus Moessner
Institute for Communication Systems (ICS)
University of Surrey, UK
Email:{adnan.akbar, f.carrez, k.moessner}@surrey.ac.uk

Ahmed Zoha
Qatar Mobility Innovation Center (QMIC)
Doha, Qatar
Email: ahmedz@qmic.com

***Abstract*—The widespread use of IoT devices has opened the possibilities for many innovative applications. Almost all of these applications involve analyzing complex data streams with low latency requirements. In this regard, pattern recognition methods based on CEP have the potential to provide solutions for analyzing and correlating these complex data streams in order to detect complex events. Most of these solutions are reactive in nature as CEP acts on real-time data and does not exploit historical data. In our work, we have explored a proactive approach by exploiting historical data using machine learning methods for prediction with CEP. We propose an adaptive prediction algorithm called Adaptive Moving Window Regression (AMWR) for dynamic IoT data and evaluated it using a real-world use case. Our proposed architecture is generic and can be used across different fields for predicting complex events.**

***Index Terms*—Complex event processing, internet of things, machine learning, pattern recognition, proactive applications, regression, time series prediction**

I. Introduction

The Internet of Things (IoT) have significantly increases the number of devices connected to the Internet. This includes a variety of devices such as sensors, smart phones and increasingly soft sources such as crowd sensing or users as sensors. The availability of data generated by these diverse devices has opened new opportunities for innovative applications across different fields such as supply chain management systems [1], intelligent transportation systems [2] and smart cities [3].

Most of the IoT applications such as traffic management system or supply chain logistics of big super markets involve large data sets which have to be analyzed in near real-time for decision making. Data from different sensors in IoT is generated in the form of real-time events which form complex patterns where each complex pattern represent a unique event. These unique events must be interpreted with minimal time latency in order to apply them for decision making in the context of current situation.

The need for processing, analyzing and inferring from these complex patterns in near real-time forms the basis of a recent research area called Complex Event Processing (CEP) [4]. The Research area of CEP includes processing, analyzing and correlating event streams from different data sources to infer more complex events in near real-time. The inherent distributed nature of CEP makes it ideal candidate for many IoT applications as evident by numerous examples found in literature such as managing large traffic data [5] or providing automatic managing systems for smart buildings [6].

Although CEP provide solutions to deal with data streams in real-time but it lacks the predictive power provided by Machine Learning (ML) and statistical data analysis methods. Most of the CEP applications are intended to provide reactive solutions by correlating data streams using predefined rules as the events happen and does not exploit historical data due to its limited memory. But in many applications, early prediction of an event is more useful than detecting it when it has already occurred. For example, it will be more useful to predict the traffic congestion as compared to detecting it. The advantage of predicting an event is more obvious if we imagine the gain of predicting Tsunami disaster as compared to detecting it after it has happened [7].

On the other hand there are several methods found in literature based on ML and statistics which have the ability to provide innovative and predictive solutions such as for predicting traffic flow [8], energy demand for buildings [9], and travel-time for passengers [10]. ML methods exploit historical data and applies diverse disciplines such as probability and artificial intelligence to train the models in order to make predictions about the future. They have the potential to provide the basis for proactive solutions for IoT applications but they lack the power of scalability and processing multiple data streams which is provided by CEP.

In our work, we exploit both approaches and combine CEP and ML in order to provide a proactive solution for IoT applications. The promise behind our work is that if the input to the CEP is predicted data, then the complex event detected by CEP using causal and temporal pattern recognition techniques will be a predicted complex event. In contrast to the current prediction methods which are based on static model parameters, we propose an adaptive prediction algorithm called Adaptive Moving Window Regression (AMWR) for dynamic IoT environments, which utilizes moving window for training the model and updates the model as new data arrives. It tracks the error and prevent it from propagation. The size of the training window is found automatically which optimize the performance for specific dataset and the size of prediction window is also adaptive in nature in order to ensure certain accuracy in the prediction.

In short, following contributions are made in this paper.

978-1-5090-0366-2/15/$31.00 ©2015 IEEE

- We propose a framework based on ML and CEP for predicting complex events for proactive IoT applications.
- We propose an adaptive prediction algorithm for dynamic IoT data streams
- We evaluated our method on a real-world data set and show that our proposed algorithm achieves an accuracy upto 90 %.

The remainder of the paper is organized as follows. Section II summarizes recent work done using CEP and ML for IoT applications. Section III explains our proposed architecture along with the description of different components involved for the implementation of our algorithm. We have demonstrated the feasibility of our solution by implementing a prototype and evaluating the results on a real-world use-case scenario in Section IV. Finally we conclude the paper and highlight our future work in section V.

II. Related Work

There are many examples found in literature where CEP is utilized in IoT for innovative applications by correlating different data streams to infer complex events. One such example is found in [5] where authors propose to use CEP for traffic monitoring. They used the data from different sensors such as loop detectors, radars and cameras and from the traffic infrastructure elements such as traffic lights to detect complex events such as congestion or accidents. Another example is given in [6] where authors developed a distributed architecture using CEP for managing lighting systems and monitoring of devices for malfunctioning.

In recent years, there are few research efforts which have explored the possibility of combining predictive analytics (PA) methods such as ML and statistics with CEP to provide proactive solutions. Initially, it was proposed in [11], where authors presented a conceptual framework for combining PA with CEP to get more value from the data. Although, the idea of combining both technologies was encouraging but they did not support their work with any practical application. Another example is given in [12] where authors used probabilistic event processing network to detect complex events and then used these complex events to train multi layered Bayesian model for predicting future events. Their proposed prediction model uses Expectation Maximization (EM) algorithm [13] which is an iterative optimization algorithm with high computational cost. Its complexity increases exponentially as the training dataset increases thus making it unsuitable for large scale IoT applications. They demonstrated their approach on the simulated traffic data with the assumption of availability of statistical data of vehicles which is very unlikely to be available in a real-world use-case.

In [14], authors provide a basic framework for combining time series prediction with CEP for monitoring of food and pharmaceutical products in order to ensure their quality during the complete cycle of supply chain. The authors highlighted the open issues related to prediction component such as model selection and model update as new data arrive but did not address these issues and left it for their future work. Another recent example of using time series prediction of data for CEP in order to provide predictive IoT solutions is mentioned in [15] where authors implemented neural network for prediction. They demonstrated their approach on the traffic data and used 60 days of data to train the neural network. One major drawback with this approach is that it cannot track the error in predictions and in case of erroneous readings, error will propagate and will keep on increasing.

In addition to the examples mentioned above, there are many applications found in literature where time series prediction using ML methods were addressed individually such as for predicting energy demands [9], traffic flow [16] and meteorological data [17]. Traditional methods for predicting time series used fix models with large amounts of historical data for training. The performance of these models may deteriorate over time as the statistical properties of the underlying data may changes with time due to concept drift [18]. More recent and advanced time series prediction methods address these issues as evident by the examples given in [17], [19] and [20] where different variants of moving window were proposed. Most of these methods are application dependent and lack a generic solution for applying to different domains.

III. Proposed Architecture

The proposed architecture illustrating our approach is shown in Figure 1. Data from different sources is accessed over a RESTful web service before applying AMWR on it. Afterwards, predicted data is published on the message bus which is a distributed pub/sub architecture under specific topics from where CEP system can access it.

More details about the different components involved in our approach are described below.

A. Adaptive Moving Window Regression (AMWR)

Adaptive Moving Window Regression (AMWR) is the first and foremost step in our proposed architecture for proactive IoT applications. We propose and developed an adaptive prediction algorithm called AMWR for dynamic IoT data. In general, prediction models are trained using large historical data and once the model is trained it is not possible to update the model. In real time dynamic environments, the performance of the model may deteriorate over time due to change in statistical properties of underlying data. The context of the application may change resulting in the degradation of prediction model performance. For such scenarios, we propose a prediction model which utilizes moving window of data for training the model and once new data arrive, it calculates an error and retrain the model accordingly. The optimum size for training window is found graphically and is specific to the underlying data stream. Our proposed approach is adaptive in nature as it tracks the error and prevents it from propagating by retraining the model periodically. The size of prediction window or forecast horizon is also adaptive which is derived by the performance of the model in order to ensure a certain reliability in the prediction. The flowchart of overall approach

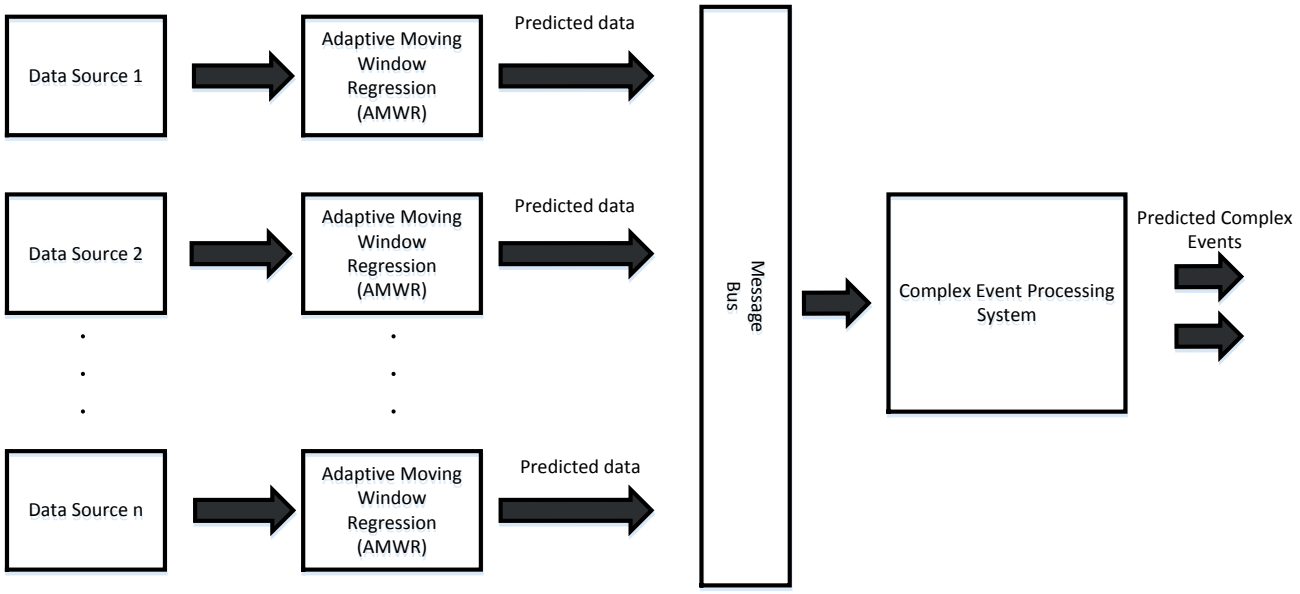

Fig. 1. Proposed Architecture and Block Diagram

is shown in the Figure 2. There are three main steps involved in the implementation of AMWR which are described below.

1) Selection of regression algorithm
2) Finding optimum training window size
3) Adaptive prediction window

1) Selection of Regression Algorithm: Traditionally, statistical methods like ARMA and ARIMA [21] were used for time series regression but recently the trend is shifted towards more sophisticated ML models such as different variants of Support Vector Regression (SVR) and Artificial Neural Networks (ANN) because of their robustness and ability to provide more accurate solutions. We have implemented our approach using SVR due to its ability to model non-linear data using kernel functions. SVR is an extension of SVM which is widely used for regression analysis [22]. The main idea is the same as in SVM, it maps the training data into higher feature space using kernel functions and find the optimum function which fits the training data using hyper plane in higher dimension.

2) Optimum Training Window Size: The choice of optimum training window size is bit tricky. In general, the accuracy of prediction model increases as the size of training data increases. It reflects to have large historical data for training prediction models so that it covers all possible patterns spanning time series. Although this approach generates generic and accurate model for prediction but there is one major drawback associated with it. If the behavior or statistics of the underlying data changes, trained model is unable to track the changes and result into erroneous readings. Parameters for the model are fixed and it is unable to take the error into consideration.

In contrast to this approach, we observed that many time series data follows a certain pattern and the near feature

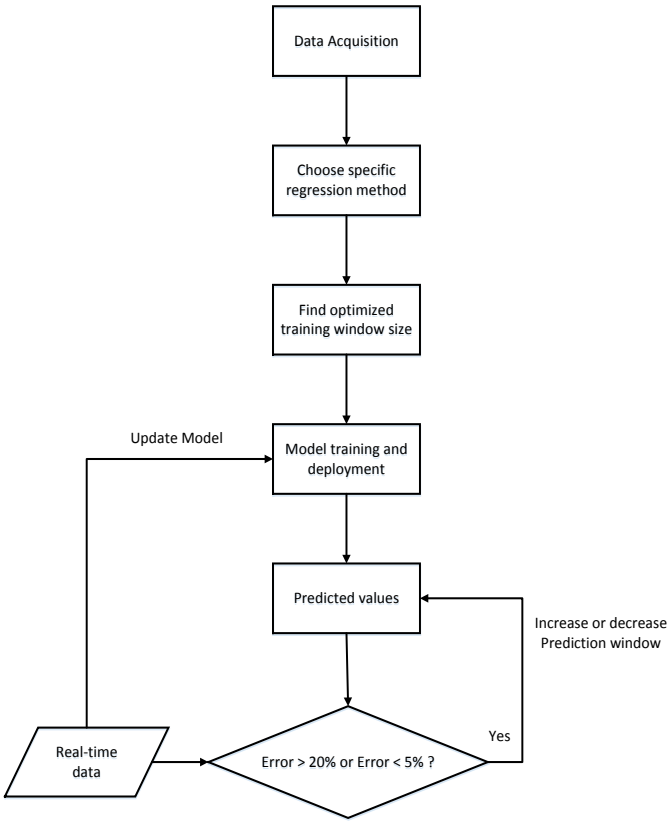

Fig. 2. Flowchart for Adaptive Moving Window Regression

readings are more dependent on the recent historical data. For example, traffic state does not go bad from good at once, instead it follows the certain pattern which indicates that traffic is becoming denser. We used this observation to

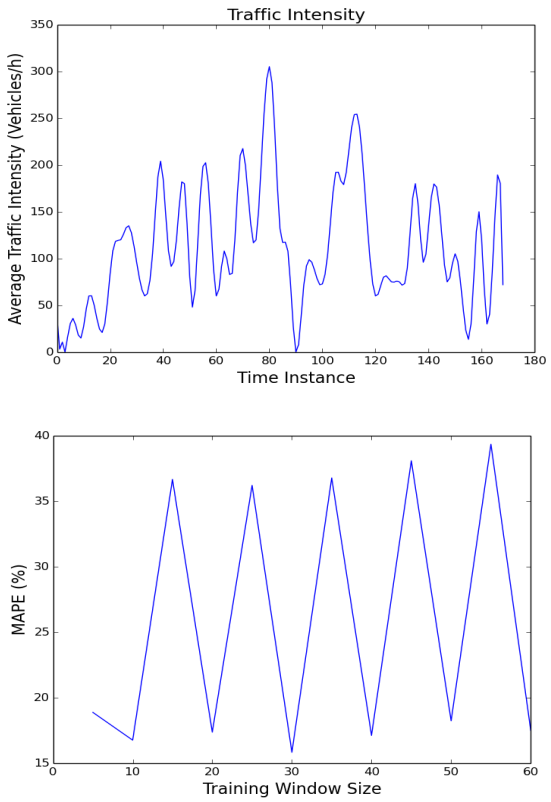

Fig. 3. a) Raw traffic intensity data b)Training window size vs MAPE

propose a small moving window for training the prediction model. The size of training window was found graphically as shown in the Figure 3 which shows the plot of different training window sizes against Mean Absolute Percentage Error (MAPE) for traffic intensity (average number of vehicles per hour). MAPE is an accurate metric for evaluating the performance of prediction models and can be calculated as:

$$MAPE(\%) = 1/n \sum_{t=1}^{n} |(\frac{Y_t - Y_t^{'}}{Y_t})| \times 100 \qquad (1)$$

where Y_t represents actual data, $Y_t^{'}$ represents predicted data and n represents the total number of predicted values. The optimum size of training window is chosen which minimizes the MAPE over historical data. It can be seen from the Figure 3 that error increases and decreases periodically with the training window size. The reason behind is that the raw data has a periodic pattern and for a certain size of training window, it spans the complete period and can predict the local trend accurately. But as the size of training window increases, it introduces over fitting and resulting into increased error. Our algorithm finds the optimum training window size automatically during training phase which depends on the input data.

3) Adaptive Prediction Window: In our work, we propose to have an adaptive size for prediction window in order to ensure a certain level of accuracy. The intuition behind our approach is to increase the size of prediction window if the accuracy of model is high and decrease it if the performance of prediction model decreases. The performance of the model is evaluated by comparing the predicted data with the actual data when it arrives. Algorithm 1 shows our approach for adaptive prediction window.

Algorithm 1 Adaptive Prediction Window Size

function PREDICTIONWINDOW(y_{act}, y_{pred})
  $MAPE = mean(abs((y_{act} - y_{pred})/y_{act}) * 100)$
  if $MAPE > 20\%$ **then**
    $PredictionWindow = PredictionWindow - 1$
  else if $MAPE < 5\%$ **then**
    $PredictionWindow = PredictionWindow + 1$
  else
    $PredictionWindow = PredictionWindow$
  end if
  return $PredictionWindow$
end function

B. Complex Event Processing

The main objective of CEP is to provide the processing capability of big data engines which enables it to analyze the data and extract patterns on the run in real time in a distributed manner. The major difference from the traditional big data engines was that CEP can handle multiple events which are seemingly unrelated and can correlate them to provide a desired and meaningful output. CEP combines data from different sources aggregate and correlate it using different components which forms the basis of every CEP engine. These components are briefly explained below:

1) Filters: Event filtering is the most basic functionality which supports other more complex patterns. Not every event is of interest for the consumers and a user might be interested in only specific events. Lets consider a simple example of a temperature sensor which generates a reading every second; If a user is only interested in the temperature greater than a specified threshold, it is defined in the filter. Then if only the conditions becomes true, the event would be published to the observer. Typical conditions consist of equals or greater/less than etc.

2) Windows: Windows provide a tool to extract temporal patterns from the incoming events to infer a complex event. The two most basic type of windows are:

a) Time Window: It enables to define a time window to extract events lying only in that window. The temporal relation between different events plays an important role in evaluating complex event. For example 5 degree centigrade temperature change in a room in 1 hour will have different meaning as compared to the same temperature change in 1 minute. The former observation can be resulting from the heater being switched on and later might be caused by a fire. The time

window can be a fixed time window or a sliding time window. Simple arithmetic tasks like finding maximum, minimum or aggregated value also require the definition of time window.
b) Tuple Window: In contrast to time window, tuple window acts on the number of events defined. Aggregation of every five samples is a typical example of tuple window operation.

3) Joins: The functionality of Joins is to correlate events from different streams using simple logical operations such as and, or. For example if we have the data from a temperature sensor and smoke sensor, a more complex event fire can be derived as

If temp $>$ Threshold **and** Smoke is True; generate complex event **Fire**.

4) Aggregation: Most of the CEPs have built in aggregation functions such as sum, min, max and count to assist for simple analysis. Some more advanced platforms also offer statistical parameters such as standard deviation and variance. Aggregation functions always require a window to assign.

C. Example Rule for CEP

Different combination of rules can be applied to infer temporal and casual pattern from multiple data streams using components described above. Pseudo code for one such example for inferring bad traffic state or congestion is described in the algorithm 2. In this example, we assume traffic speed and traffic intensity data streams as inputs. CEP detects an event when the average traffic speed and average traffic flow is less than the threshold values which are defined by the users and then checks if the values are continuously decreasing with respect to previous recorded values. If the pattern indicates that traffic state is becoming worse, it generates a complex event of congestion. Now if the input is predicted data as in our approach, the complex event detected will also be in the future. This is just one example in order to demonstrate how CEP rules can be exploited to find more complex events.

Algorithm 2 Example Rule for CEP

1: **for** $(speed, intensity) \in TupleWindow(3)$ **do**
2: **if** $(speed(t) < speed_{thr}$ and $intensity(t) < intensity_{thr}$ AND
3: $speed(t+1) < speed(t)$ and $intensity(t+1) < intensity(t)$ AND
4: $speed(t+2) < speed(t+1)$ and $intensity(t+2) < intensity(t+1))$ **then**
5: Generate complex event Congestion
6: **end if**
7: **end for**

IV. Experimental Evaluation

In order to evaluate the proposed method, we have used the traffic data provided by city of Madrid. The city of Madrid has deployed thousands of traffic sensors at fixed locations for measuring several traffic features including average traffic intensity (number of vehicles per hour) and average traffic speed which are direct indicatives of traffic state. They help the city administrators to understand the traffic conditions and take reactive measures in order to ensure smooth traffic.

TABLE I
MAPE (%) for traffic Speed and traffic Intensity

Training Window	Traffic Speed	Traffic Intensity
10	34.35	16.75
15	16.88	36.66
20	**10.59**	17.36
25	36.45	36.20
30	14.55	**15.82**
35	11.17	36.77
40	38.08	17.11

We applied our proposed prediction algorithm on traffic speed and traffic intensity data in order to predict in future. Real-time data is accessed over RESTful web service [1] in xml format. We have used historical data of few hours which is available online [2] to find the optimum size of training window. Table I shows the error calculation for traffic speed and traffic intensity with different training window sizes. Our algorithm selects the window size corresponding to least error during training phase. After deployment, as new data arrive, it calculates the error by comparing it with predicted data and as error starts to increase, it decreases the prediction window size and retrain the model in order to track the actual data. The size of training window is quite small which enables the system to work in near real-time.

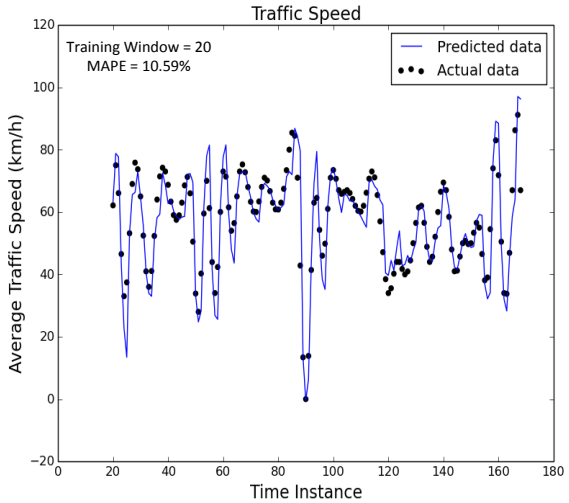

Fig. 4. Prediction Result for traffic speed using AMWR

Figure 4 shows the result of the prediction algorithm on average traffic speed readings during a day. As it can be

[1] http://informo.munimadrid.es/informo/tmadrid/pm.xml
[2] http://datos.madrid.es/portal/site/egob/

seen, that predicted values are tracking the actual data quite accurately. The reason behind it is that if there is an error in the predictions, it is incorporated and model is updated accordingly and hence it prevents the error from propagating. The size of the training window used was 20 corresponding to least prediction error of 10.59 %.

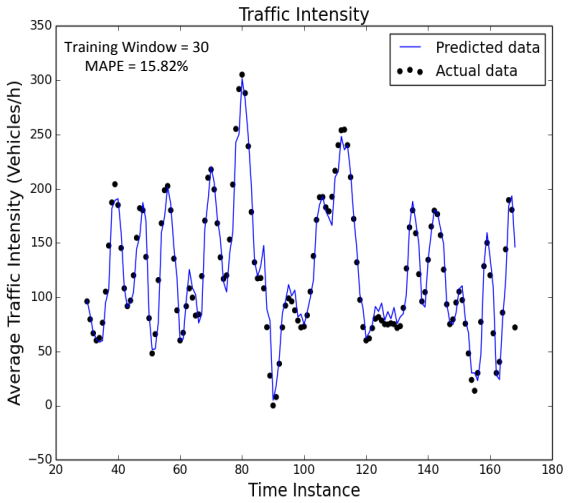

Fig. 5. Prediction Result for traffic intensity using AMWR

Similarly, Figure 5 shows the results for the prediction algorithm on average traffic intensity. The optimized size of moving window was 30 for this feature set which result into 15.5% error as can be seen in table I.

Once we have the predicted values for traffic speed and traffic intensity, CEP can be used to infer traffic state using the rules mentioned in algorithm 2. Event detected by CEP will be in future providing enough time for traffic administrators to manage traffic pro-actively and avoiding congestion before it happens.

V. Conclusion and Future Work

In this paper, we have demonstrated that ML methods can be used in conjunction with CEP in order to provide proactive solution for IoT applications. Our proposed method is adaptive in nature and is able to cope with dynamic environments as opposed to current state of the art methods. We evaluated our proposed solution on a real-world use case and demonstrated its potential use for intelligent transportation systems. Different data streams have different prediction error and this error propagates as data streams are combined to infer a complex event. In future, we will work on the modeling of our system in order to demonstrate the error propagation in the system. We also aim to test our approach on high velocity data as well as on other IoT scenarios.

Acknowledgments

The research leading to these results was supported by the EU FP7 project COSMOS under grant ICT-609043.

References

[1] B. Yan and G. Huang, "Supply chain information transmission based on rfid and internet of things," in *Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on*, vol. 4. IEEE, 2009, pp. 166–169.

[2] L. Xiao and Z. Wang, "Internet of things: A new application for intelligent traffic monitoring system," *Journal of networks*, vol. 6, no. 6, pp. 887–894, 2011.

[3] A. Zanella, N. Bui, A. P. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, 2014.

[4] O. Etzion and P. Niblett, *Event processing in action*. Manning Publications Co., 2010.

[5] B. Tarnauca, D. Puiu, D. Damian, and V. Comnac, "Traffic condition monitoring using complex event processing," in *System Science and Engineering (ICSSE), 2013 International Conference on*. IEEE, 2013, pp. 123–128.

[6] C. Y. Chen, J. H. Fu, T. Sung, P.-F. Wang, E. Jou, and M.-W. Feng, "Complex event processing for the internet of things and its applications," in *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1144–1149.

[7] Y. Tsuchiya and N. Shuto, *Tsunami: Progress in prediction, disaster prevention and warning*. Springer Science & Business Media, 2013, vol. 4.

[8] J. Rzeszótko and S. H. Nguyen, "Machine learning for traffic prediction," *Fundamenta Informaticae*, vol. 119, no. 3-4, pp. 407–420, 2012.

[9] A. Ahmad, M. Hassan, M. Abdullah, H. Rahman, F. Hussin, H. Abdullah, and R. Saidur, "A review on applications of ann and svm for building electrical energy consumption forecasting," *Renewable and Sustainable Energy Reviews*, vol. 33, pp. 102–109, 2014.

[10] X. Fei, C.-C. Lu, and K. Liu, "A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1306–1318, 2011.

[11] L. J. Fülöp, Á. Beszédes, G. Tóth, H. Demeter, L. Vidács, and L. Farkas, "Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics," in *Proceedings of the Fifth Balkan Conference in Informatics*. ACM, 2012, pp. 26–31.

[12] Y. Wang and K. Cao, "A proactive complex event processing method for large-scale transportation internet of things," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014.

[13] T. K. Moon, "The expectation-maximization algorithm," *Signal processing magazine, IEEE*, vol. 13, no. 6, pp. 47–60, 1996.

[14] S. Nechifor, B. Tarnauca, L. Sasu, D. Puiu, A. Petrescu, J. Teutsch, W. Waterfeld, and F. Moldoveanu, "Autonomic monitoring approach based on cep and ml for logistic of sensitive goods," in *Intelligent Engineering Systems (INES), 2014 18th International Conference on*. IEEE, 2014, pp. 67–72.

[15] B. Thomas, F. Jose, s. Jordi, A. Almudena, and T. Wolfgang, "Real time traffic forecast," *Atos scientific white paper*, vol. 2013, 2013.

[16] A. Ding, X. Zhao, and L. Jiao, "Traffic flow time series prediction based on statistics learning theory," in *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*. IEEE, 2002, pp. 727–730.

[17] Y. Suzuki, H. Ibayashi, Y. Kaneda, and H. Mineno, "Proposal to sliding window-based support vector regression," *Procedia Computer Science*, vol. 35, pp. 1615–1624, 2014.

[18] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[19] M. Vafaeipour, O. Rahbari, M. A. Rosen, F. Fazelpour, and P. Ansarirad, "Application of sliding window technique for prediction of wind velocity time series," *International Journal of Energy and Environmental Engineering*, vol. 5, no. 2-3, pp. 1–7, 2014.

[20] G. Lee, U. Yun, and K. H. Ryu, "Sliding window based weighted maximal frequent pattern mining over data streams," *Expert Systems with Applications*, vol. 41, no. 2, pp. 694–708, 2014.

[21] W. W.-S. Wei, *Time series analysis*. Addison-Wesley publ, 1994.

[22] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.