Event correlation and forecasting over high-dimensional streaming sensor data

Vassilis Papataxiarhis
Dpt. of Informatics and Telecommunications
National and Kapodistrian University of Athens
Athens, Greece
vpap@di.uoa.gr

Stathes Hadjiefthymiades
Dpt. of Informatics and Telecommunications
National and Kapodistrian University of Athens
Athens, Greece
shadj@di.uoa.gr

Abstract—Event management in sensor networks is a multidisciplinary field involving several steps across the processing chain. In this paper, we discuss the major steps that should be performed in real- or near real-time event handling including event detection, correlation, prediction and filtering. We explain the rationale behind each considered step and we focus on an event correlation algorithm based on a variable-order Markov model. The proposed theory is applied on the maritime domain and is validated through extensive experimentation with real sensor streams originating from large-scale sensor networks deployed in a maritime fleet of ships.

Keywords—event correlation; event forecasting; sensor networks; change detection

I. Introduction

In a Sensor Network (SN), sensor nodes constitute the network components that collect data about the environment within which they are deployed. Sensing elements capture contextual information to support context-aware applications from diverse domains (e.g., maritime, intelligent transportation, smart farming). Being based on the collected information, a SN intends to detect events that take place during the network lifetime and may depict, or even affect, the state of the system. The term 'event' is used to describe an alteration on one or more variables monitored by the system (we call them *context attributes - CA*). In this paper, we follow a stepwise approach to analyze the CA and unveil the transitions between the different system states that govern the operation of the network. Such interconncetions are normally hidden under the streams of raw data captured by the system.

The contribution of this study is multifold. First, we show that the determination of events in sensor networks can be accommodated within a change detection framework in the context of multivariate time series. Then, we propose an event correlation scheme based on the idea of partial matching [1] by essentially implementing a variable-order Markov model for the correlation of multivariate event data. Finally, we discuss a framework for modeling event dependencies through a probabilistic temporal logic programming paradigm and we propose a methodology to deal with outdated dependencies.

II. Motivation

In a SN, streams mostly arrive as sequences of raw data that provide instant measurements or summaries (e.g., mean, max) regarding observed phenomena that may change over time. Due to its frequency, raw data are usually of limited value even for system experts that intend to reach a higher level of understanding for the internal dynamics of the system (e.g., expected sequences of events in the near future, faulty situations, abnormalities). The sensor streams are handled and analyzed in an online fashion in order to identify times when the probability distribution of the respected time series changes. Each sensor stream is transformed into a binary stream that represents time points of unexpected behaviour of a CA *(event detection)*. Event streams can provide refined information of the data that may be of high importance to the decision makers that need to act proactively in order to sustain the proper behaviour of the system. For example, the abnormal increase of the values reported by a smoke detector could essentially be an evidence of a fire. As a next step, we focus on the derivation of a dependency structure that describe time-dependent regulations between the incoming data streams and illustrate possible transitions on the system state *(event correlation)*. The resulted rules are extracted from dependency structures that are constructed in near real-time to represent interdependencies among the measured variables *(event prediction)*. The derived dependencies that result in separate time steps are filtered in order to balance between outdated patterns and event sequences of previous steps that could still be of importance *(event filtering)*. Fig. 1 presents the workflow for the online processing of events originated from sensor networks as it is presented in the following sections.

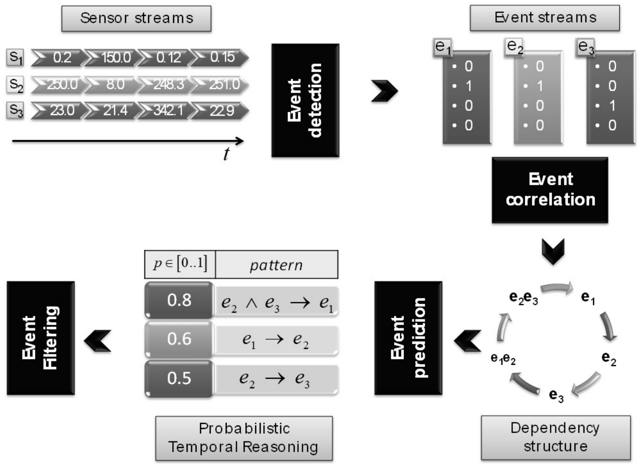


Fig. 1. Online processing of events in sensor networks.

III. Prior Work

From a research perspective, literature provides a wide range of approaches for the detection and correlation of events in real-time systems. In [2], the authors propose a probabilistic framework for the detection of abnormalities in streaming logs that come from multiple event sources. The proposed scheme assumes the events already detected and the identification of abnormalities is based on the derivation of a directed probabilistic graph that represents the historical footprint of the event occurrences. The graph vertices are specified by the prior probabilities for the occurrence of event types while each directed edge comes with a conditional probability between the connected event types. The authors also propose a metric to quantify the similarity between two event streams. A similar graph structure is presented in [3] with the edges corresponding to the joint probabilities of two event types (i.e., bidirectional graph). The authors apply information theory measures to provide a historical analysis of abnormality events. They also provide metrics for the determination of root-cause events as well as events that can, possibly, lead to a system crash. Both frameworks do not take into account sequences of correlated events since the derived conditional and joint probabilities, respectively, refer to events occurred during the same time window, but not in subsequent time steps. For example, consider a case of an event stream where whenever an event of type A occurs at time *t*, an event of type B also occurs at time step *t+1*. Similarly, implications imposed by joint occurrences of events are not considered due to time complexity (e.g., event types A and B have to occur both in order to result in the occurrence of event type C). In [4], the authors propose a probabilistic scheme for the correlation of distributed events in the network security field. Specifically, they apply a HMM [5] and Kalman filtering [6] to unveil spatial and temporal correlations among the observations. Kalman fitering and CUSUM algorithm [7] are adopted to detect changes in [8].

From an industrial perspective, a lot of studies have been devoted in the area of *condition-based predictive maintenance* [9] for the prediction of faults and errors in sensor networks. Most approaches are based on existing statistical methods to identify alerting situations due to abrupt changes of the probability distributions of the observed parameters [10]. In [11], the authors discuss the problem of rare events prediction in multidimensional data and they propose two approaches that address the problem. The first approach focuses on degradation detection (i.e., abnormal system behavior) that is based on one-class Support Vector Machines model [12]. The proposed method formulates a quadratic program that detects anomalies by minimizing the distance of each new multidimensional data vector with regard to a training dataset. The second approach exploits a regularized logistic regression classification method where the predictor function consists of a transformed linear combination of explanatory variables. In [13], the authors propose one model-based and one data driven method for predicting faults in multi-sensor systems. The former approach is based on an autoregressive moving average model that is realized as a Neyman-Pearson hypothesis test [14]. The second approach exploits an HMM and the Viterbi algorithm [15] to estimate the most likely sequence of hidden states. A common drawback of proposed methodologies is their validation via simulations that use synthetic data.

IV. Methodology

A. Event detection based on context changes

We observe in real-time, with some uniform frequency, multivariate time series of quantitative system performance parameters. Let s_i be a *sensor stream* reporting numeric sensor values and let $s_i(t)$ be the value of stream s_i at time t, $t \geq 0$. Assuming that n sensor streams are synchronized to report their values periodically, we represent the set of multidimensional contextual information at each time t by the *context vector (CV)*, $CV_t = (s_1(t), s_2(t), ..., s_n(t)) \in \Re^n$. Practically, each sensor stream formulates a univariate time series while the CV stream represents a multivariate time series. Event detection aims to determine the values $s_i(t)$ that constitute abrupt changes within a CV stream. Specifically, we transform each CV of length n into a binary vector of the same length with each value representing a possible change in the respective stream. We call such deviations of the normal behavior *events* and the binary vector *event vector (EV)*. Let $I = \{e_1, ..., e_n\}$ be the set of the considered event types e_i with regard to the respective sensor streams s_i, $i \in [1, n]$. An event can be any observation that does not conform to an expected pattern. An event vector at time t is represented by $EV_t = (e_1^t, e_2^t, ..., e_n^t) \in \{0,1\}^n$, where $e_i^t = e_i(t)$ is the binary value representing whether an abnormal behavior took place for stream s_i (value '1') or not (value '0').

The transformation of a CV into an EV is based on change detection algorithms that aim to identify abnormal deviations of the current values with respect to the values arrived in previous steps In the context of this work, we have experimented with algorithms that consider each sensor stream separately (*univariate change detection*), namely CUSUM algorithm [7] and Shewhart control charts [16]. The cumulative sum (CUSUM) algorithm [7] attempts to detect a change on the distribution of a time series with respect to a target value at real-time by calculating positive and negative changes and comparing these values against a positive and a negative threshold (Figure 2). CUSUM assumes that the time series follow a normal distribution. On the other hand, Shewhart control charts [19] provide a statistical measure to detect abrupt shifts by calculating the distance from the current mean value of the numerical stream (Figure 3).

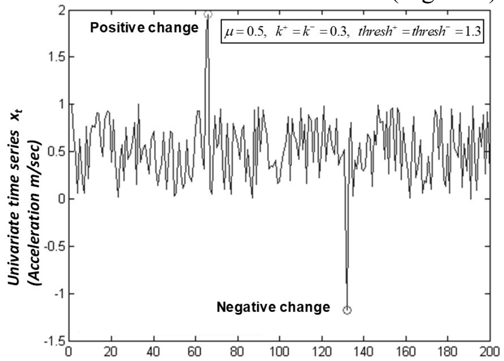


Fig. 2. Change detection based on CUSUM

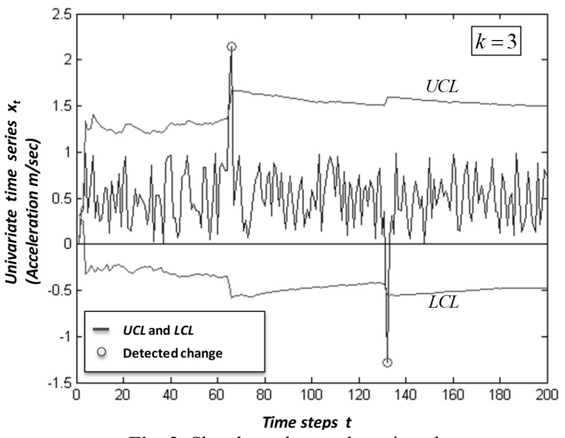


Fig. 3. Shewhart change detection chart.

B. Variable-order event correlation

Over the past decades, a lot of effort has been devoted in the correlation of event data originated from IT infrastructures in order to extract patterns that depict the internal dynamics of systems and act proactively. Typical event correlation schemes that operate over univariate time series assume the following:

- A transition from object (i.e., event or event sequence) *A* to object *B* occurs iff *B* occurs right after *A* (i.e., not within a time window).
- Only one object is considered at each step (i.e., there are no objects occurring at the same time).

However, in the case of multivariate sensor data, there is no guarantee that events would happen one at a time. On the contrary, an alerting situation or a malfunctioning system is expected to lead to several events triggered at the same time step. For example, consider the case of a fire that is monitored through multiple smoke detectors, temperature and humidity sensors. Most sensors are expected to experience a significant drift in the probability distribution of their reported values, thus leading to multiple events at the same time steps. In addition, since events occur rarely when no alerting situation takes place, the probability of having no events should be considered.

In this paper, we address the above challenges by representing the occurrence of event vector streams as a stochastic process. We propose a *variable-order correlation scheme* based on the idea of *partial matching*, which had initially been applied for web prefetching. The original partial matching algorithm [1] faces the problem of predicting the upcoming l URL accesses of a client based on information about the past m steps. The algorithm maintains a data structure that keeps track of the past URL sequences of length up to $m+l$. The algorithm determines all subsequences in the history structure that match any suffix of the last m accesses. A cut-off probability threshold p_{thr} is used to discard URLs with low probability. All items that match these suffixes exceeding p_{thr} are prefetched.

We propose a variable-order event correlation algorithm by adapting the idea of partial matching to the context of multivariate streaming sensor data. Our algorithm implements the variable-order Markov model idea where Markov chains of orders $1,...,m$ are combined to predict event sequences of length up to m. We consider the following parameters:

(a) Number of past multivariate event vectors considered (m). It represents the *maximum order* of the model.

(b) Number of steps that the algorithm would predict in the future (l).

Let $I=\{e_1,...,e_n\}$ be the set of the considered event types e_i with regard to the respective sensor streams s_i, $i\in[1,n]$. We represent the n-dimensional event vector at time t by $EV_t=\left(e_1^t,e_2^t,...,e_n^t\right)\in\{0,1\}^n$, where $e_i^t=e_i(t)$ are the binary values resulted from the change detection step and they represent whether an abnormal behavior took place for stream s_i (value '1') at time t or the value $s_i^t=s_i(t)$ that arrived was in the expected range.

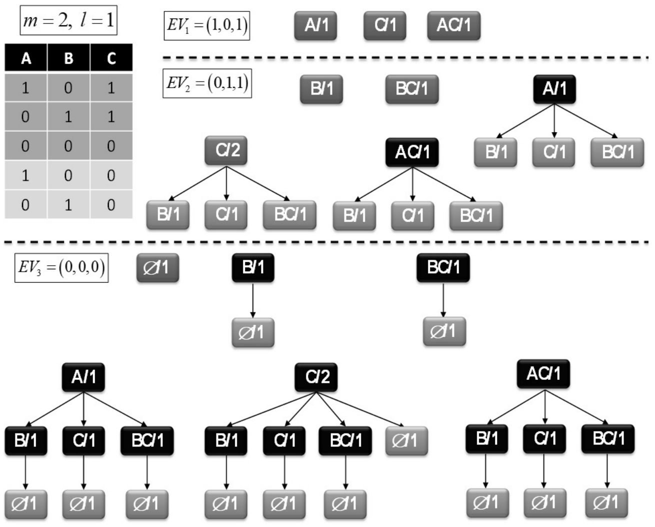
The algorithm operates in an online fashion by maintaining a data structure that keeps track of the past sequences of events of length up to $m+l$. The considered data structure is a collection of trees $T=\left\{T_{I_r}\right\}$, $I_r\in\mathbf{P}(I)$, instead of multiple graphs, in order to conserve space since common prefixes of multiple sequences are stored only once. The index $I_r\subseteq I$ of each tree represents the event subset related to the root node of the tree. At each time step t where a new event vector $EV_t=\left(e_1^t,e_2^t,...,e_n^t\right)\in\{0,1\}^n$ arrives, the algorithm updates the data structure so that all event sequences of length up to $m+l$ are included. Hence, the height of each tree is $h_{\max}=m+l-1$ at most. Each *tree node* v can be represented as a tuple of the form:

$$v=\left\langle I_v,N_v^t\right\rangle,\ I_v\subseteq I$$

where I_v is a specific set of events related to this node and N_v^t is the frequency of the observed pattern starting from the root node until the current time step t. At each step, the frequencies referring to sequences of the last $m+l$ steps are updated. Contrary to the stepwise correlation algorithm, the nodes represent combinations of events that occurred at the same time step, not states of all possible events realizations. Hence, multiple nodes may correspond to the same time step. The reason behind this decision is the practical need of keeping different frequencies for the same sub-patterns that occur within bigger event patterns (e.g., sub-pattern BC may occur within patterns ABC and CBC).

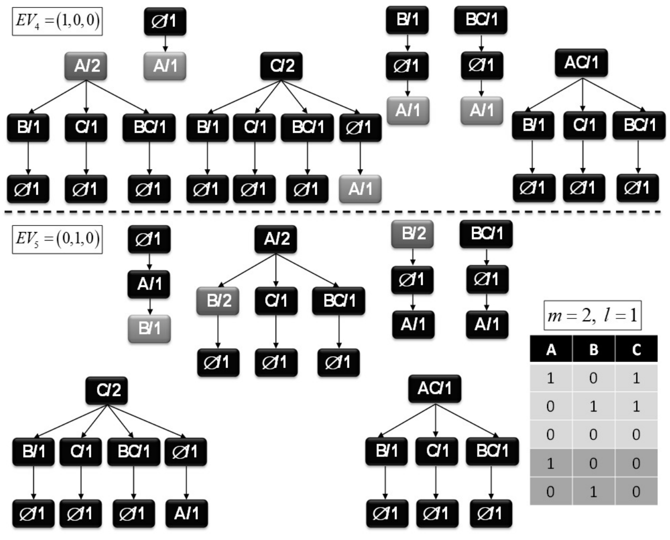
An indicative example of the algorithm with $m=2$ and $l=1$ is presented in Fig. 4 (steps 1-3) and Fig. 5 (steps 4-5). The event vectors refer to three types of events (A, B and C). At each step, dark gray rectangles illustrate root nodes for new trees that have to be constructed. Black rectangles represent nodes that remain unchanged with respect to the previous step. New leaf nodes are highlighted in light gray. The number close to each node denotes the frequency of the observed pattern starting from the root of the tree.

The prior probability of an event set I_r of a root node equals to the frequency N_r^t of the root node divided by the current time step and is represented by $P_r^t = \frac{N_r^t}{t}$. For example, the prior probability of the event C in step 5 equals to $P_C^5 = \frac{2}{5}$. A path within a tree represents a sequence of events that has been occurred and it is annotated with a probability value. The probability of such an event pattern of the form $r, \ldots, u, v$ (i.e., path within a tree starting from root node r) equals to the frequency of the node v at the current time step divided by the frequency of the node u one time step before and is represented by $P_{r,\ldots,u,v}^t = \frac{N_v^t}{N_u^{t-1}}$. For example, the probability of pattern $A \to B \to \varnothing$ after time step 5 equals to $P_{A,B,\varnothing}^5 = \frac{N_\varnothing^5}{N_B^4} = 1$.



A	B	C
1	0	1
0	1	1
0	0	0
1	0	0
0	1	0

Fig. 4. Variable-order event correlation example – Steps 1-3.



A	B	C
1	0	1
0	1	1
0	0	0
1	0	0
0	1	0

Fig. 5. Variable-order event correlation example – Steps 4-5.

The variable-order correlation algorithm assumes, in the worst case, that all possible combinations of n different types of events must be kept at each step. In that case, the number of possible trees in the worst case is $\sum_{i=1}^{n} \binom{n}{i} = 2^n$. However, storing all possible states is of little use in practice since the probability of occurrence of a big number of events at the same time step is practically very low. If we consider only combinations consisting of k events at most (with k fixed and independent of n), then the upper limit on the number of the considered nodes is significantly reduced to the following estimation:

$$|V| = \sum_{i=0}^{k} \binom{n}{i} = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} \ldots + \binom{n}{k} = 1 + n + \frac{n \cdot (n-1)}{2} + \ldots + \frac{\prod_{j=0}^{k-1} (n-j)}{k!(n-k)!} = O\left(n^k\right)$$

This means that, assuming that combinations of k event types at most (with k fixed) will occur, the time complexity of the algorithm becomes polynomial. Since the event frequencies can be updated cumulatively, the algorithm is able to construct its history structure on the fly by continuously updating it for every new event vector arrived. Additionally, under the same assumption, the maximum number of nodes included in each tree (i.e., in case that all trees are full and complete) is

$$\sum_{i=1}^{h_{\max}} \left(\sum_{j=1}^{k} \binom{n}{j} \right)^i = \frac{1 - n^{k(l+m-1)}}{1 - n^k} = O\left(n^{m+l-1}\right)$$

where $h_{\max} = l + m - 1$ is the maximum height of each tree. Practically, this means that the total number of nodes that need to be updated in each step for all the trees is polynomial ($O\left(n^{k+m+l-1}\right)$), given the fact that the algorithm will consider combinations of k event types at most. Since the event frequencies can be updated cumulatively, the algorithm is able to construct its history structure on the fly by continuously updating it for every new event vector arrived.

C. *Event prediction based on PTL*

The ability to formally express the dependencies among multivariate event data and reason over the different system states over time is of high importance in order not only to provide accurate predictions about future system behavior but also to facilitate the supervision of the system by experts. The variable-order event correlation approach can be adequately modeled using the probabilistic temporal logic (PTL) programming paradigm [17] as a tool to formulate the temporal dependencies among the event patterns extracted by the event correlation schemes and reason over time. Specifically, we represent the occurrence of events of type A_i at time step t by *time formulae* A_i^t. The *probabilistic temporal formulae* consisting of atoms A_i^t with probability p are represented by A_i^t / p. Consequently, time dependencies between events are captured through *probabilistic temporal rules* of the form:

$$A \to B : [t, p]$$

where A, B are formulae consisting of ground atoms and typical logic programming operators for conjunction, disjunction and negation ($\wedge, \vee, \neg$, respectively) while the formula B is annotated with a probability value p and a time unit t . Intuitively, such rule states that whenever the formula A included in the *body* of the rule stands true, the *head* formula B will be also true with probability p *after* t time units. This type of expressions allows us to represent logical operations over atoms that result in the probabilistic ground truth of other atoms within specific time.

D. Adaptive event filtering

Most sensor networks are highly dynamic by nature since they target to capture changes within non-static environments and spaces. Hence, the dependencies resulted in each step could be altered over time. A weakness of typical event correlation methods is their inefficiency to identify *outdated dependencies*. For example, rules extracted from a correlation scheme based on Markov-models could reflect dependencies between events that took place a long time ago and do not conform to the recent activity of events. On the contrary, a memoryless scheme (e.g., one based on a sliding window) takes into account only the very recent activity by completely forgetting past event occurrences.

A simple, yet effective, approach for answering the question of when an extracted rule (i.e., dependency) becomes too old is to apply a time-dependent *aging* or *decay function*. In general, aging functions constitute time-based forgetting mechanisms. The main rationale in the proposed adaptation strategy is that a drift on the behavior of a data stream should be a gradual process. Also, rules that have been extracted recently are most of the times more important and valid than the old ones. In the same context, the importance of a rule should decrease over time.

Here, we represent an aging function by $\lambda(r_t) = f(t)$, where r_t is a rule extracted t time steps before. At each time step, each rule r_t is assigned with an aging value (*weight*) depending on how old the rule is. The weight of each rule represents the importance of the rule at the current time step. In general, different $f(t)$ may be used depending on the particular situation for which events are being processed. Here, we discuss a linear and an exponential approach.

A linear aging function scheme with $\lambda(r_i) = -\frac{2k}{n-1}(i-1)+k+1$ is presented in [18] where n is the number of the considered past steps, i is a counter starting from the current step and goes back over time , r_i is a rule derived i time steps before and $k \in [0,1]$ is the percentage of decreasing the weight of a rule by each step. By varying k the slope of the aging function can be adjusted. Similarly, an exponential aging scheme with $\lambda(r_i) = \exp(-ki)$ is presented in [19]. The parameter k controls how fast the weights decrease over time. For larger values of k , less weight is assigned to the rules. If $k = 0$ all the steps are assigned with the same weights.

Fig. 6 and 7 depict the above linear and exponential aging functions with distinct k values.

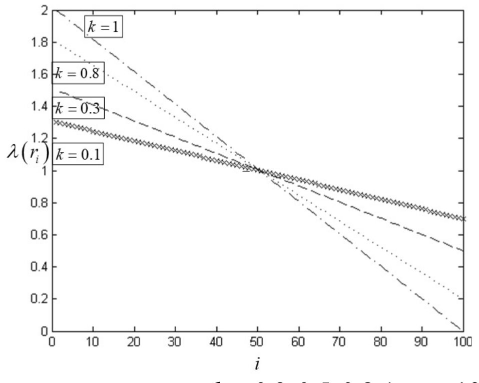


Fig. 6. Linear aging, $k = 0.3, 0.5, 0.8, 1$, $n = 100$.

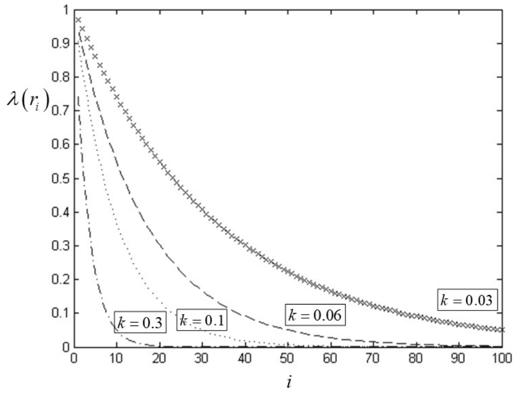


Fig. 7. Exponential aging, $k = 0.03, 0.06, 0.1, 0.3$, $n = 100$.

If a rule that represents exactly the same dependency is generated in multiple time steps, the proposed framework balances between the, possibly varying, probabilities of rules by weighting the respective probability values according to the weights of the rules over time. Specifically, we monitor the rules extracted within the last Δt_{mem} time steps and we use the following weighted form to estimate the probability of a rule derived within this time interval:

$$p_{r,\Delta t} = \frac{\sum_{i=t-\Delta t+1}^{t} \lambda(r_i) \cdot p_{r,i}}{\sum_{i=t-\Delta t+1}^{t} \lambda(r_i)}$$

Consider a case where $\Delta t_{mem} = 3$ and a rule r that was extracted before 2 time steps with probability $p_{r,t-2} = 0.8$ and at the current time step with probability $p_{r,t} = 0.3$. Following Eq. (3) and taking into account the linear aging function with $k = 0.8$, the rule will be, finally, assigned with a probability

$p_{r,\Delta t} = \frac{(0.8 \cdot 1) + (0.3 \cdot 1.8)}{2.8} = 0.47$. This way, a memory-less correlation scheme can be equipped with memory capabilities within a time interval Δt_{mem}.

V. Experimental Evaluation

All the experiments were performed by taking advantage of real data coming from the maritime domain. Specifically, our scheme has been assessed in real-world scenarios where the machinery and the infrastructure of three (moving) ships were constantly monitored making this information available for further analysis and processing at the back-end in real-time. The monitored sensor streams provided more than 3.5MB of data per minute resulting in a total of more than 450GB in less than 3 months (for all the three ships). The considered parameters included ship speed, acceleration, direction, wind speed, machine temperature and humidity, person count, gas percentage, GPS, water levels and water inclination.

CUSUM and Shewhart algorithms were used to detect potential abnormalities in the sensor streams. For each numeric stream, the result of the change detection process was a binary stream that was provided as input to the event correlation algorithm. We performed several tests to measure the precision of the proposed Markov chain based on the two change detection algorithms. Each precision value was measured in 8000 sequential time steps. At each step, the predictions for the current time step have been checked of whether they became valid after the considered period of time and the precision value was updated accordingly.

Fig. 8, 9 and 10 demonstrate the performance of the proposed forecasting scheme over different probability thresholds of the derived rules over time. The probability threshold value is an indication of the strength of each rule. Specifically, values close to '1' indicate that the rule was, in general, valid in the recent past while values close to '0' indicate obsolete rules. Hence, whenever a rule derived during the event prediction and filtering phases comes with a probability lower than this threshold, this rule has been discarded due to its low strength. Otherwise, it is inserted in the pool of rules that are evaluated based on their accuracy of predictions.

A first and clear result that can be deduced from the figures is that Shewhart detection dominates over CUSUM algorithm in all cases. The main reason behind this result is that CUSUM assumes that the underlying probability distribution of the dataset is a normal one. This is not the case for Shewhart which makes no assumptions with respect to the dataset. As a result, CUSUM fails to provide accurate detections that can lead to increased levels of precision with respect to future event forecasting. Apart from that, Fig. 8 depicts that CUSUM also fails to comply with high probability thresholds since the overall precision of results decreases as the probability threshold increases. On the other hand, as the probability threshold increases, the system improves its performance in terms of predictions in the case of Shewhart change detection, as it was expected.

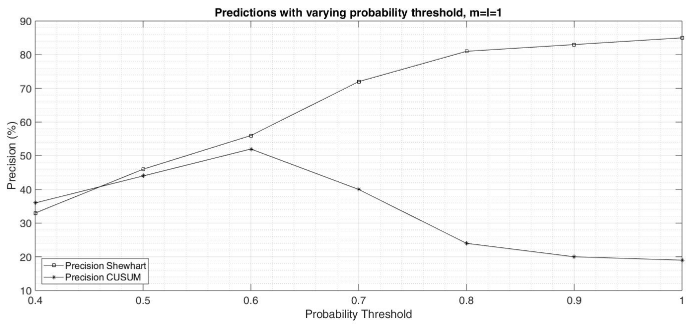


Fig. 8. Predictions with varying *probability threshold*, *m*=*l*=1.

Similarly to Fig. 8, Fig. 9 demonstrates the capability of the system to predict events that will occur two steps beyond based on events occurred during the last time step only. Compared to the case where the predictions referred only to the next step (Fig. 8), the quality of results is obviously lower (i.e., lower levels of precision). For instance, in the Shewhart case of Fig. 9, the precision value ranges between 24.8% and 58.1% while the respective curve in Fig. 8 ranges between 33.2% and 85.8%. Again, results are better in the case of Shewhart detection. This becomes more clear in Fig. 10 (*m*=2, *l*=1) where the precision results based on CUSUM detection do not exceed 12.3% in any of the considered cases.

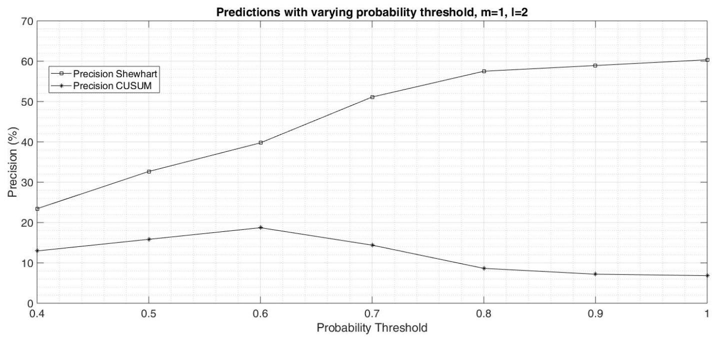


Fig. 9. Predictions with varying *probability threshold*, *m*=1, *l*=2.

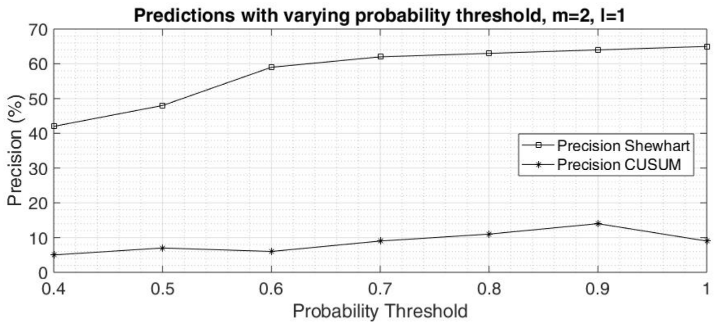


Fig. 10. Predictions with varying *probability threshold*, *m*=2, *l*=1.

We also assessed the quality of future event predictions based on different values of the fixed combinations parameter (i.e., *k*) that was considered each time. Fig. 11 demonstrates the precision of the event predictions under different values of *k*, *l*, and *m* when Shewhart algorithm was used to detect changes in the streams. No aging function was adopted in the experiment and the probability threshold was fixed to 0.6. It becomes clear that the accuracy of results is higher in the case where the event correlation algorithm is based only on the events that took place in the last step and not on the previous ones. Specifically, the 70.2% of predictions of the 'next step' events were valid when *k*=1 while the results became worse as the *k* value was decreased (i.e., *k*=2,3,4,5). It was also true that a higher number of steps considered to support a prediction (i.e., higher *m* values) do not necessarily result in more accurate predictions. In almost all cases, we measured lower precision values as the *k* value was increasing. This practically means that most of the events can be deduced by taking into consideration only a single event that took place in the previous time step.

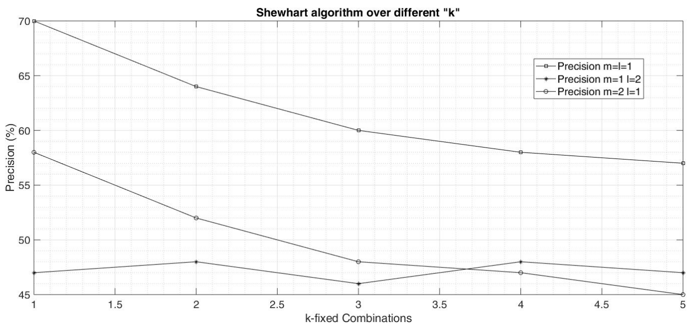


Fig. 11. Predictions with varying *k* based on Shewhart detection.

Similarly, we assessed the performance of the predictions using the CUSUM algorithm for feeding the event correlation scheme (Fig. 12). As expected, the results were worse in the general case compared to the ones acquired using the Shewhart algorithm. This can be easily explained by the fact that CUSUM assumes a normal probability distribution for the underlying numeric stream and, again, none of the real-world datasets complied with this requirement. As a result, the precision values were much lower than in the Shewhart case. For instance, all the precision results in the case when the two last steps were considered in order to predict a single step fall down to less than 25% of accuracy. The curve that was affected less by the underlying change detection algorithm was the one with *m*=*l*=1. Again, no aging policy was adopted and the probability threshold was set to 0.6.

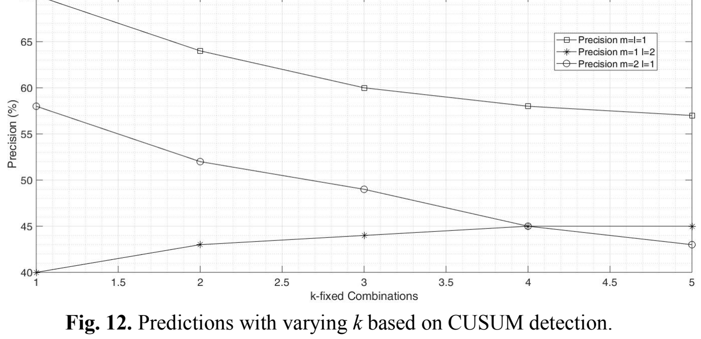


Fig. 12. Predictions with varying *k* based on CUSUM detection.

Finally, we assessed the performance of the aging functions for both detection schemes considered over time. Figures 13 and 14 present the system performance in the case where linear and exponential aging, respectively, was adopted. In both cases, it seems that the overall accuracy of predictions is enhanced compared to the case where no aging function is applied. Once more, Shewhart detection performed better than CUSUM in both cases because of the abnormality in the probability distribution of the data. Specifically, the overall precision on the prediction in the case of Shewhart detection was always more than 75% while it did not exceed the value of 44% in the CUSUM case. Moreover, Shewhart detection performed better in linear aging than it did in exponential aging. On the other hand, CUSUM performed better in exponential aging than in linear aging. In all cases the *k* value does not seem to significantly affect the overall quality of results. The probability threshold was set to 0.7 while *m*=*l*=1.

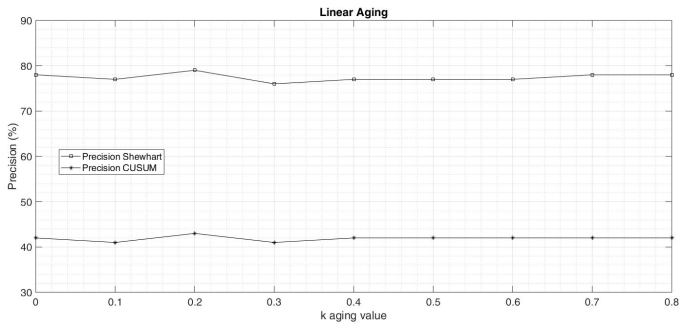


Fig. 13. Predictions with varying *k* based on linear aging.

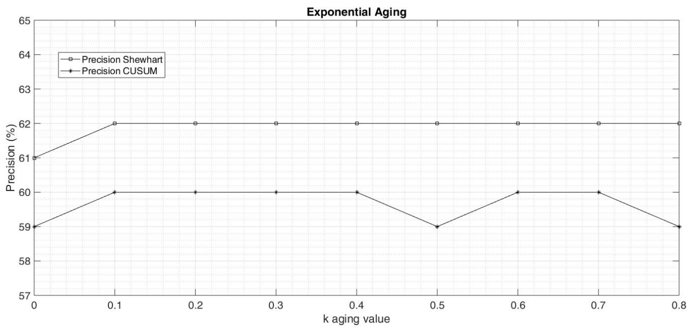


Fig. 14. Predictions with varying *k* based on exponential aging.

VI. Conclusion

The paper discussed several parts of the processing chain in the context of online event management over sensor data. First, we demonstrated that change detection techniques can be used for the determination of events in sensor networks. Then, we proposed an online scheme for correlating multivariate event data. The algorithm involved a variable-order Markov model to capture sequences of multiple events. We also showed that representation of dependencies in order to predict future event sequences can be accommodated within a probabilistic temporal knowledge representation framework that allows the formulation of probabilistic rules. To address the important issue of outdated dependencies, we have set-up a time-dependent framework for filtering the extracted rules over time through aging factors. The experimental evaluation of the presented approaches was based on real-world data from the maritime domain.

The presented correlation scheme faced the problem of correlating multivariate event data by, essentially, adopting high-order Markov models to represent frequent event patterns. Since these models capture sequences of events that take place in subsequent steps, they have certain limitations in representing direct dependencies among events that occur within a larger timeframe. In most cases correlated events may not happen one after the other since the effects of the causal may need time to be observed. For example, we can consider the case where an event of type A which is coupled with a temperature sensor occurs exactly ten time steps after the occurrence of event type B related to a smoke detector. In that case, even if a ten-order model could capture this dependency through multiple time steps, it could not provide a direct correlation between the cause (i.e., increase of temperature) and the effect (i.e., existence of smoke). As a next step, we plan to investigate approaches that can address time dependencies among events within larger timeframes. A possible alternative could take advantage of a sliding window that would add memory to the algorithm.

Evidence of a bad or malfunctioning behavior of a sensor network is often embedded within sequences of events detected in distributed nodes across the network. Such state of importance may involve multiple time steps in order to reveal its effects. Also, several bad states could lead to the similar types of events. Our future research agenda will focus on the identification of hidden (i.e., unobservable) system states that may affect the observable variables (i.e., sensors). Hidden Markov Models set up a modeling framework for the representation of such causalities and the determination of unobservable states of a system with regard to the observed measurements.

Another aspect of importance is the improved characterization of detected changes in the context of a time series. Most of the existing change detection algorithms do not provide further annotation of the detected novelties (i.e., 0/1 characterization). We plan to extend existing univariate change detection techniques in order to provide multi-level detection of changes for numerical attributes. This step involves the identification of novel value ranges (i.e., clusters) with new entry values that do not comply with the existing clusters.

ACKNOWLEDGEMENT

This work has been implemented in the context of the ROBORDER project [20]. ROBORDER has received funding from the European Union's Horizon 2020 Research and Innovation programme under grant agreement No. 740593.

REFERENCES

[1] L. Fan, P. Cao, W. Lin, and Q. Jacobson. 1999. Web prefetching between low-bandwidth clients and proxies: potential and performance, in the *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, Pages 178-187 ACM New York, NY, USA ©1999, DOI: 10.1145/301453.301557.

[2] A.N. Harutyunyan, A.V. Poghosyan, N.M. Grigoryan, and M.A. Marvasti, "Abnormality analysis of streamed log data", in IEEE/IFIP Netwotk Operations and Management Symposium (NOMS), 5-9 May, Krakow, Poland, 2014.

[3] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, "An anomaly event correlation engine: Identifying root causes, bottlenecks, and black swans in IT environments", VMware Technical Journal, vol. 2, issue 1, pp. 35-45, 2013.

[4] G. Jiang, and G. Cybenko. 2004. Temporal and Spatial Distributed Event Correlation for Network Security, In the Proc. of *2004 American Control Conference*, pages 996 – 1001, Vol. 2 Boston, June 30 - July 3.

[5] O. Cappe, E. Moulines, and T. Ryden. 2005. *Inference in Hidden Markov Models*, 2005:Springer-Verlag.

[6] R. E. Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering* **82** (1): 35–45. doi:10.1115/1.3662552

[7] E. S. Page. 1954. Continuous Inspection Scheme. Biometrika 41 (1/2): 100–115. doi:10.1093/biomet/41.1-2.100. JSTOR 2333009.

[8] M. Severo and J. Gama. 2006. Change Detection with Kalman Filter and CUSUM, In the *Proc. Int'l Conf. Discovery Science*, pp. 243-254, 2006.

[9] J. Veldman, H. Wortmann, W. Klingenberg. 2011. Typology of condition-based maintenance . *Journal of Quality in Maintenance Engineering* , *17* (2), 183 - 202. 10.1108/13552511111134600.

[10] M. Basseville, and I. V. Nikiforov. 1993. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1993).

[11] S. Alestra, C. Bordry, C. Brand, E. Burnaev, P. Erofeev, A. Papanov, and D. C. Silveira-Freixo. 2014. Rare event anticipation and degradation trending for aircraft predictive maintenance, In the *Proceedings of 11th World Congress on Computational Mechanics (WCCM XI),* July 21, 2014.

[12] I. Steinwart and A. Christmann. 2008. *Support Vector Machines*. Springer, 2008.

[13] A. Marjanovic, G. Kvascev, P. Tadic, Z. Durovic. 2011. Applications of Predictive Maintenance Techniques in Industrial Systems, *Serbian Journal of Electrical Engineering Vol. 8, No. 3*, November 2011, 263-279, DOI: 10.2298/SJEE1103263M.

[14] K. Fukunaga. 1990. *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, USA, 1990.

[15] A. J. Viterbi .1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory 13 (2): 260–269*. DOI:10.1109/TIT.1967.1054010.

[16] D. Montgomery. 2005. *Introduction to Statistical Quality Control*. Hoboken, New Jersey: John Wiley & Sons, Inc. ISBN 978-0-471-65631-9. OCLC 56729567.

[17] P. Shakarian, A. Parker, G. I. Simari, V. Subramanian, Annotated probabilistic temporal logic. *ACM Transactions on Computational Logic*, 12:14:1–14:44, 2011.

[18] I. Koychev. 2000. Gradual forgetting for adaptation to concept drift. In Proceedings of ECAI Workshop Current Issues in Spatio-Temporal Reasoning, Berlin, Germany, pp. 101{106. ECAI Press.

[19] R. Klinkenberg. 2004. Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis 8 (3), 281-300, 2004.

[20] Autonomous swarm of heterogeneous robots for border surveillance, https://roborder.eu/.