Journal of The Korea Society of Computer and Information
Vol. 24 No. 2, pp. 139-147, February 2019
www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.02.139

Automatic False-Alarm Labeling for Sensor Data

Taufik Nur Adi*, Hyerim Bae**, Nur Ahmad Wahid***

Abstract

A false alarm, which is an incorrect report of an emergency, could trigger an unnecessary action. The predictive maintenance framework developed in our previous work has a feature whereby a machine alarm is triggered based on sensor data evaluation. The sensor data evaluator performs three essential evaluation steps. First, it evaluates each sensor data value based on its threshold (lower and upper bound) and labels the data value as “alarm” when the threshold is exceeded. Second, it calculates the duration of the occurrence of the alarm. Finally, in the third step, a domain expert is required to assess the results from the previous two steps and to determine, thereby, whether the alarm is true or false. There are drawbacks of the current evaluation method. It suffers from a high false-alarm ratio, and moreover, given the vast amount of sensor data to be assessed by the domain expert, the process of evaluation is prolonged and inefficient. In this paper, we propose a method for automatic false-alarm labeling that mimics how the domain expert determines false alarms. The domain expert determines false alarms by evaluating two critical factors, specifically the duration of alarm occurrence and identification of anomalies before or while the alarm occurs. In our proposed method, Hierarchical Temporal Memory (HTM) is utilized to detect anomalies. It is an unsupervised approach that is suitable to our main data characteristic, which is the lack of an example of the normal form of sensor data. The result shows that the technique is effective for automatic labeling of false alarms in sensor data.

▸Keyword: False-alarm detection, False-alarm labeling, anomaly detection, hierarchical temporal memory

I. Introduction

According to ANSI/ISA-18.2 [1], an alarm system is a device combining hardware and software to detect an alarm state, communicate the indication of that state to operators, and record any change in it. Alarms are recognized as an effective tool for early detection of process upset [2] as well as identification of near misses followed by appropriate actions necessary to bring a process back to normal operation [3]. In short, an alarm system plays a vital role in ensuring operational safety and efficiency in most industrial sectors.

An alarm system also acts as a layer of protection preventing escalation of an abnormal event that otherwise could result in catastrophe [4]. The design of the protection layer(s) in an industry depends on its industrial process. Fig. 1 shows the different layers of protection associated with a typical industrial process. A properly functioning alarm system should provide relevant abnormality-related information to the operator to enable that person to perform the necessary intervention in avoiding catastrophic incidents.

• First Author: Taufik Nur Adi, Corresponding Author: Hyerim Bae
*Taufik Nur Adi (taufiknuradi@pusan.ac.kr), Industrial Engineering Department, Pusan National University
**Hyerim Bae (hrbae@pusan.ac.kr), Industrial Engineering Department, Pusan National University
***Nur Ahmad Wahid (nawa410@pusan.ac.kr), Big Data Department, Pusan National University
• Received: 2019. 01. 28, Revised: 2019. 02. 19, Accepted: 2019. 02. 19.
• This work was supported by a 2-Year Research Grant of Pusan National University

Many industrial incidents have been recorded as evidence of poor alarm system performance. For instance, during the investigation of the Milford Haven refinery explosion in 1997, the Health and SafetyExecutive (HSE) found that one of the major causes of the incident was the generation of too many alarms, or in other words, an alarm flood (275 alarms in 11 minutes), prior to the explosion [5] [6]. In another incident known as BP Texas City, the alarm system failed to warn the operator about an unsafe condition in the tower, which failure led to an explosion and fire leaving 15 dead 180 others injured.

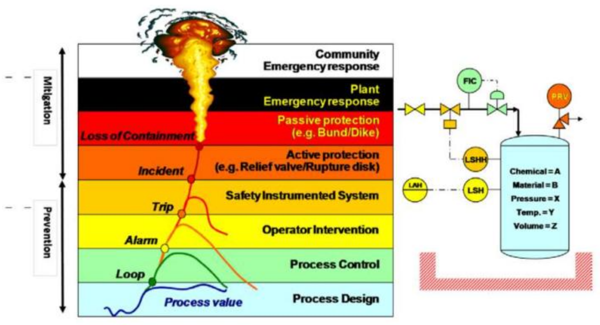


Fig. 1. Layers of protection [4]

Such incidents imply that effective and robust alarm systems can improve the safety of working environments and that poor alarm systems reduce the ability of an operator to take necessary action in abnormal situations.

Some of the significant symptoms associated with an ineffective alarm system are:

1. Inappropriate or no master alarm database;
2. No operator action required under an alarm condition
3. No clear guidelines or specifications for adding or deleting alarms;
4. Poor alarm testing procedures and records;
5. Operating procedures not written in consideration of alarms;
6. Changes in alarm settings during shift changeovers;
7. Important alarms being missed during accidents;
8. Minor upsets resulting in large numbers of alarms that operator cannot keep up with;
9. Alarms appearing for a considerable amount of time (even 24 hours) or alarms being activated even when there is no upset condition;
10. Too many high-priority alarms.

If more than three of these symptoms are in evidence, action to improve the design of an alarm system is indispensable. Meanwhile, a past study identified five specific non-design-related alarm-system problems requiring attention and further investigation [7]. One of them is the handling of nuisance alarms. According to the ANSI/ISA-18.2 [1], the definition of a nuisance alarm is an alarm that annunciates excessively, unnecessarily, or does not return to normal after the correct response is taken. Nuisance alarms, also known as false alarms, require attention because they not only incur trouble during normal operation, but also are among the main factors contributing to excessive operator workload. Ineffective alarm management caused by false alarms can lead to severe loss. Hence, there is a critical need to develop a technique to deal with false alarms.

Fig. 2 shows the architecture of the cloud-based predictive maintenance system developed in our previous work [17]. The predictive maintenance system receives sensor data from a third-party application that collects it from machine(s) on the shop-floor within a manufacturing operation.

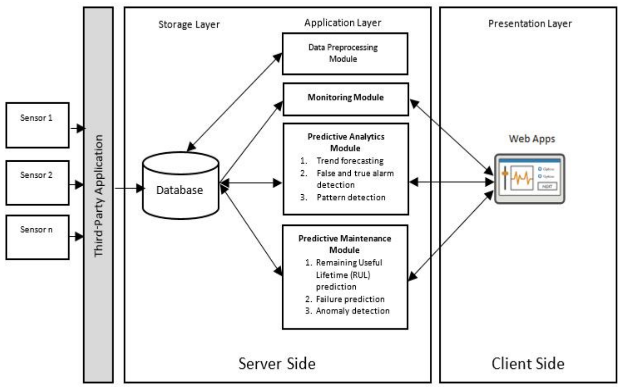


Fig. 2. Architecture of Cloud-based Predictive Maintenance [17]

The sensor data is stored in the database before undergoing processing by a particular module in the application layer. In alarm-system terms, the data processing module has two essential functions for evaluation of sensor data: evaluation of sensor data based on a threshold, and calculation of the duration of alarm occurrence for initial detection of a potential alarm. By relying only on this evaluation technique, however, our system suffers from unnecessary alarm overloading. Therefore, the result obtained from this data processing module still needs to be manually assessed by the domain expert to determine whether the alarm is false or true. Another problem now arises, this one related to false-alarm assessment. Simply, the vast amount of sensor data to be assessed manually by the domain expert renders this task overly prolonged and inefficient. In our case study, the rule for determining a false alarm was known; hence, mimicking how the domain expert assessed the sensor data was possible. Proposing a method for automatic labeling of sensor data as a false alarm is the focus of this paper.

This paper is organized as follows. Section 2 discusses the related work, section 3 outlines the proposed method, and section 4 presents the system implementation and results. Finally, section 5 draws conclusions.

II. Preliminaries

2. Related works

2.1 False-alarm detection

There have been many studies related to false-alarm detection. [8] proposed a false-alarm detection architecture for cyber-physical systems designed for healthcare applications. It had been determined that utilization of a wide range of medical sensors generated a large number of false alarms that tended to confuse medical staff and reduce the efficiency of overall healthcare delivery. The implementation of the threshold alarm method as combined with multiple classifiers in the decision set effectively improves alarm performance in regard to both accuracy and efficiency. [9] attempts to reduce the false alarm ratio of installed Fire Detection and Fire Alarm Systems (FDAS) in Germany. Installed FDAS are useful means to achieve the objectives of fire protection engineering which are to increase the life safety and property protection. The highly sensitive sensor in fire detectors enables fire identification in an early stage, but this also makes FDAS susceptible to false alarms. The effectiveness of FDAS is critical to the firefighter operation, hence the understanding of false alarm to reduce its ratio is important. In the computer science field, research on the Intrusion Detection System (IDS) [10] applied a layered filtering approach to anomaly detection in order to reduce false-alarm rates. The idea behind the use of layered filtering is inspired by the process of air or water pollution elimination. Each filter layer is responsible for eliminating a specific pollution type. Similarly, each industry produces different characteristics of sensor data. By using the same concept, a layered filter could help to distinguish between normal and anomalous behavior of sensor data.

2.2 Anomaly detection

An anomaly is a state, condition or behavior that deviates from the norm. Anomaly detection refers to the identification of anomaly patterns in data. Anomaly detection is also known as outlier detection. Related applications ranging from military surveillance [11] to health care [12] have been widely studied. Generally, anomaly-detection strategies and algorithms can be grouped into two categories: statistical techniques and machine-learning methods. The principle of the statistical techniques is the creation of a statistical model derived from normal behavior data and its application to a statistical inference test to determine if a data instance fit the model or not. The data is labeled as anomalous if the data instances have a low probability of being generated from the learned model. As for machine-learning methods, anomaly detection requires labeled data that denote instances as normal or anomalous.

The sensor data utilized in our present case study was considered as time-series data, since its data points were indexed chronologically. It should be noted that there are some challenging factors that need to be considered when performing anomaly detection for time-series data [11]:

1. The pattern of the data keeps changing. Therefore, the anomaly-detection algorithm for time-series data should be able to learn continuously from the evolving data.
2. The data always contains temporal dependencies; thus, considering the temporal context in the anomaly-detection algorithm is important to the provision of more accurate prediction.
3. The anomaly-detection algorithm should perform robustly in eliminating the influence of noise.

In [13] and [14], it was concluded that neither statistical techniques nor machine learning can effectively handle the challenges thrown up by time-series data. Therefore, both studies proposed the use of Hierarchical Temporal Memory (HTM), which offers the ability to learn data patterns continuously without the need of significant manual intervention.

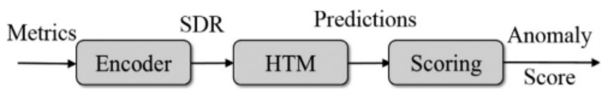


Fig. 3. HTM Framework [13]

Fig. 3 provides a brief illustration of the HTM framework. The metric values are fed as input to the HTM. These values will be converted to a sparse distributed representation (SDR) before the learning process starts. SDRs give useful attributes such as generalizability across data stream types, strong resistance to noise, and the attachment of semantic meaning to data points [13]. In the next step, the HTM learning algorithm receives a sequence of SDRs as input. During the learning process, HTM performs patterns recognition in order to derive the associations between them. Every time a new pattern is learned, it will replace the old patterns as a form of continuous learning. HTM makes a prediction based on stored patterns. If a new data instance arrives, the scoring part compares the prediction with the input and outputs an anomaly score. This score is an evaluation of the degree of the derivation.

As discussed in this paper, we combined the known rule that is used by the domain expert (explained in detail in section 3) and the anomaly-detection algorithm using HTM to effectively determine false alarms in our sensor data. Since we had no adequate sample of anomalous sensor data, we considered that the unsupervised type of HTM algorithm was suitable for our case study.

III. Proposed Method

3. Automatic false-alarm labeling framework

In this section, we discuss our proposed method for automatic false-alarm labeling and provide a detail explanation of the supporting module involved.

Fig. 4 shows the framework of automatic false-alarm labeling. The framework consists of four essential modules: the data preprocessing module, the alarm occurrence duration filter, the anomaly detector, and false-alarm labeling. Basically, our proposed method adopts the layered filter concept mentioned in section 2. Each module involved in the automatic false-alarm labeling framework applies a specific rule for filtering of sensor data. As shown in Fig. 4, the first module, the data preprocessing module, filters the sensor data based on the threshold values (upper and lower bounds).

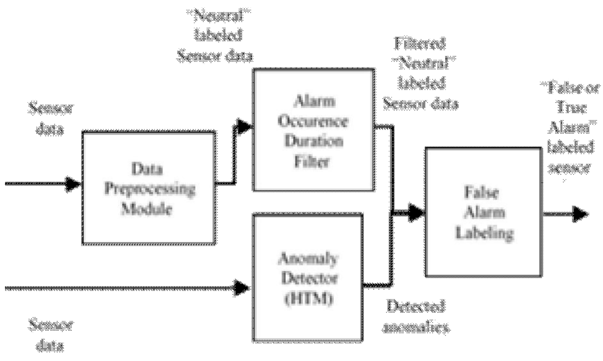

"Neutral" labeled Sensor data
Sensor data
Data Preprocessing Module
Alarm Occurence Duration Filter
Filtered "Neutral" labeled Sensor data
"False or True Alarm" labeled sensor
False Alarm Labeling
Anomaly Detector (HTM)
Detected anomalies
Sensor data

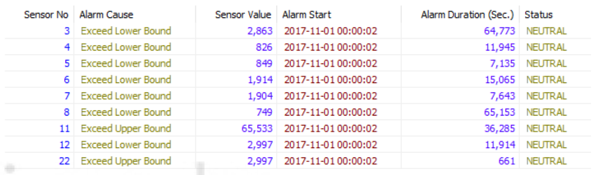
Fig. 4. Automatic false-alarm labeling framework

The second module filters the sensor data received from the first module based on the alarm occurrence duration (i.e. passing only sensor data with an alarm occurrence duration > 60 secs).

The third module, the anomaly detector, attempts to identify and distinguish between normal and anomalous patterns. The last module, false-alarm labeling, receives the results from the two previous modules. By application of particular rules that will be explained further in the next subsection, this module filters the sensor data and labels it as a false alarm if it meets the specific criteria.

3.1 Data preprocessing module

According to Fig. 2, the data preprocessing module processes raw sensor data that is stored in the database. Each sensor data has a different threshold value that is defined in advance by the domain expert to determine the state of the machine. There are two threshold values, namely the upper and lower bounds, which are used by the data preprocessing module to determine potential alarms. As mentioned in section 1, this module's function is to evaluate the sensor data based on its threshold. When the threshold is exceeded, this module will log five crucial pieces of information including alarm cause, current sensor value, starting time of alarm occurrence, alarm duration, and status.



Sensor No	Alarm Cause	Sensor Value	Alarm Start	Alarm Duration (Sec.)	Status
3	Exceed Lower Bound	2,863	2017-11-01 00:00:02	64,773	NEUTRAL
4	Exceed Lower Bound	826	2017-11-01 00:00:02	11,945	NEUTRAL
5	Exceed Lower Bound	849	2017-11-01 00:00:02	7,135	NEUTRAL
6	Exceed Lower Bound	1,914	2017-11-01 00:00:02	15,065	NEUTRAL
7	Exceed Lower Bound	1,904	2017-11-01 00:00:02	7,643	NEUTRAL
8	Exceed Lower Bound	749	2017-11-01 00:00:02	65,153	NEUTRAL
11	Exceed Upper Bound	65,533	2017-11-01 00:00:02	36,285	NEUTRAL
12	Exceed Lower Bound	2,997	2017-11-01 00:00:02	11,914	NEUTRAL
22	Exceed Upper Bound	2,997	2017-11-01 00:00:02	661	NEUTRAL

Fig. 5. Example of alarm detection logs

Fig. 5 provides an example of alarm detection logs resulting from the data processing module. In the database, the label "alarm," as produced by the threshold evaluation process, refers to the "Neutral" label in the column status, as shown in Fig. 5; this label means that the current sensor data still has to be processed by another module, which, in this case, is the alarm occurrence duration filter.

3.2 Alarm occurrence duration filter

The purpose of the alarm occurrence duration filter is to find the cause of a false alarm. One of the causes of false alarms is chattering alarms, which can account for 10 – 60% of alarm occurrences. Two other alarms closely related to false alarms are fleeting and repeating alarms. Fleeting alarms have a short-time alarm duration but do not immediately repeat [1]. Repeating alarms are alarms sounding and clearing repeatedly over a period of time [15]. The alarm occurrence duration filter module attempts to filter the sensor data received from the previous module, the data preprocessing module, by using a certain threshold value related to the duration of alarm occurrence (i.e. sensor data that has an alarm duration of less than 30 seconds will not be passed). This threshold value is defined by the domain expert. The straightforward rule applied by this module is intended to reduce the number of alarm occurrences that would be categorized as chattering alarms.

3.3 Anomaly detection using HTM

For our case study, the unsupervised type of anomaly-detection algorithm was chosen, since no adequate sample of anomalously labeled sensor data was available. The selection of HTM as the anomaly-detection algorithm was based on the consideration mentioned in section 2. HTM is a neuroscience-derived machine-learning algorithm that models spatial and temporal patterns in streaming data [13]. The model receives a continuous stream of inputs as follows:

$$\cdots, x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, \cdots \quad (1).$$

HTM networks continuously learn in order to model the spatiotemporal characteristics of their input. HTM utilizes two different internal representations to perform anomaly detection. Given an input xt, the vector a(xt) is a sparse binary code representing the current input. State vector π (xt) is utilized to represent a prediction for a(xt+1). The prediction vector incorporates inferred information about current sequences. In particular, a given input will lead to different predictions depending on both the current detected sequence and the current inferred position of the input within that sequence. How well the HTM models the current data stream will influence the quality of the prediction. a(xt) and π(xt) are recomputed at every iteration but do not directly represent anomalies.

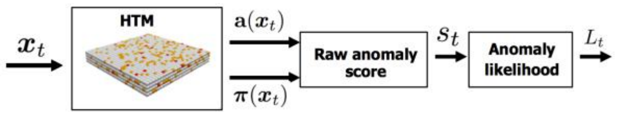


Fig. 6. Primary functional steps in HTM algorithm [25]

There are two additional steps in HTM, as shown in Fig. 6: computation of a raw anomaly score from the two sparse vectors; computation of an anomaly likelihood value that is thresholded to determine whether the system is anomalous. These additional steps will be explained in the following subsection.

3.3.1 Raw anomaly score computation

Computation of the raw anomaly score enables measurement of the deviation between the model's predicted input and the actual input. It is computed based on the intersection of the predicted and actual sparse vectors. The raw anomaly score at time t (st), is given as:

$$s_t = 1 - \frac{\pi(x_{t-1}) \cdot a(x_t)}{|a(x_t)|} \quad (2).$$

The raw anomaly score will be 0 if the current input is perfectly predicted, 1 if it is completely unpredicted, or somewhere in between depending on the similarity between the input and the prediction.

3.3.2 Anomaly likelihood computation

The raw anomaly score described above works well for predictable scenarios, but in many practical applications, inherent noise and unpredictable situations must be considered. In such situations, it is often a change in predictability that is indicative of anomalous behavior; thus, reliance only on direct thresholding of the raw anomaly score would lead to many false positives. To deal with these kinds of scenarios, rather than direct thresholding of the raw score to determine whether the current state is anomalous, it is suggested that the likelihood of anomaly be checked based on the model of the distribution of the anomaly score.

The anomaly likelihood is thus a metric defining how anomalous the current state is based on the prediction history of the HTM model. In the computation of anomaly likelihood, a window of the last W raw anomaly scores is maintained. The distribution is modeled as a rolling normal distribution wherein the sample mean and variance are continuously updated from previous anomaly scores as follows:

$$\mu_t = \frac{\sum_{i=0}^{i=W-1} s_{t-i}}{k} \quad (3).$$

$$\sigma_t^2 = \frac{\sum_{i=0}^{i=W-1} (s_{t-i} - \mu_t)^2}{k-1} \quad (4).$$

The subsequent process computes a recent short-term average of anomaly scores and applies a threshold to the Gaussian tail probability to determine whether or not to declare an anomaly. The Gaussian tail probability is the probability that a Gaussian variable will obtain a value larger than x standard deviations above the mean. The anomaly likelihood is defined as the complement of the tail probability:

$$L_t = 1 - Q\left(\frac{\tilde{\mu}_{t-} \mu_t}{\sigma_t}\right) \quad (5).$$

Where

$$\tilde{\mu}_t = \frac{\sum_{i=0}^{i=W'-1} s_{t-i}}{j} \quad (6).$$

W' is a window for a short-term moving average, where W'« W. Lt is thresholded and reported as an anomaly if it is very close to 1:

$$anomaly\ detected \equiv L_t \geq 1 - \epsilon \quad (7).$$

In clean predictable scenarios, Lt behaves similarly to st. In these cases, the distribution of scores will have a very small variance and will be centered near 0. Any spike in st will similarly lead to a corresponding spike in Lt.

3.4 False-alarm labeling

The false-alarm labeling module evaluates the sensor data received from the alarm occurrence duration filter using a particular rule combined with the result from the anomaly-detection module. As shown in Table 1, the result from the anomaly detection contains information related to the detected anomalies such as the time of the anomaly occurrence, the metric value, the three different types of anomaly level, and the raw anomaly score. From these results, we consider only the detected anomalies of "high" or "medium" value.

Table 1. Example of anomaly-detection result

Timestamp	Metric Value	Anomaly Level	Raw Anomaly Score
11/6/2017 8:08	3140.412	HIGH	0.586793832
11/6/2017 8:41	3360.438	HIGH	0.630290405
11/2/2017 16:50	4018.619	LOW	0.018347916
11/2/2017 16:51	4007.85	LOW	0.018347916

There are several rules applied for labeling of false alarms:

1. All of the alarm occurrence overlaps with anomaly occurrences of "high" and/or "medium" level will be set as false alarms regardless of the duration of the alarm occurrence.
2. All of the alarm occurrences that only overlap with anomaly occurrences of "low" level will be set as true alarms regardless of the duration of the alarm occurrence.

In other words, if high- or medium-level anomalies are detected within the alarm occurrence period, the given alarm probably was triggered by the anomaly condition, and as such, will be considered to be a false alarm.

IV. Implementation and Results

In this section, we present the system environment, the implementation of the proposed method, and the results.

4.1 System environment

All of the modules for automatic false-alarm labeling were developed in the environment specified in detail in Table 2 and using PHP technology combined with the Codeigniter framework version 3.17. For anomaly detection, we used open-source HTM Studio, the source code for which can be found in the Numenta repository [28].

Table 2. System environment of automatic false-alarm labeling

Development Environment	
Web Server	Apache Web Server 2.4
Database	Maria DB 10.2
PHP Runtime	5.6.30
PHP Framework	Codeigniter 3.1.7
Production Environment	
Operating System	Windows Server 2016 Datacenter
Web Server	Apache Web Server 2.4
Database	Maria DB 10.2

4.2 Implementation and results

We ran the application on a physical computer (specifications: Intel® Xeon® CPU E3-1220 V5 @ 3.40GHz; 16GB RAM). The experiment was conducted using a real sensor dataset retrieved for the period 2017-11-01 00:00:00 to 2017-11-07 23:59:59 (no. of instances: approx. 72,278).

The statistical information on the sensor data was as follows: min. value: 0; max. value: 4,150; average: 948,29. The threshold values for this sensor data were 2,094 and 1,047 as the upper and lower bounds, respectively. Based on the defined threshold values, the initial alarm detection performed by the data preprocessing module identified 69,890 (96,6%) of 72,278 likely alarm conditions, meaning that this 96,6% was labeled as “Neutral” for processing by the next module and determination as a false or true alarm. The occurrence duration of the identified alarm had a range of value from 2 to 86,398 seconds. The alarm duration threshold in the alarm occurrence duration filter module was set to 60 seconds, which is to say that an alarm of duration less than or equal to 60 seconds would be set as a “false” alarm. After going through the filtration process in the alarm duration filter module, the number of detected alarms from the previous module was reduced from 69,890 to 68,511 (= 1,379 or 1,97%).

Table 3. Detailed information on detected anomalies

Timestamp	Metric Value	Anomaly Level
11/6/2017 8:08	HIGH	0.586793832
11/6/2017 8:41	HIGH	0.630290405
11/6/2017 8:50	HIGH	0.524650555
11/6/2017 10:55	HIGH	0.586793832
11/6/2017 12:55	HIGH	0.6083351
11/7/2017 6:16	HIGH	0.722230663
11/7/2017 7:58	HIGH	0.524650555
11/6/2017 8:40	MEDIUM	0.414057288
11/6/2017 8:51	MEDIUM	0.485281348
11/6/2017 8:53	MEDIUM	0.449933687
11/6/2017 8:55	MEDIUM	0.431789015
11/6/2017 10:53	MEDIUM	0.414057288
11/6/2017 12:41	MEDIUM	0.414057288
11/6/2017 12:53	MEDIUM	0.431789015
11/6/2017 12:56	MEDIUM	0.431789015
11/7/2017 6:15	MEDIUM	0.485281348
11/7/2017 6:26	MEDIUM	0.485281348
11/7/2017 8:13	MEDIUM	0.466211693

Fig. 7 shows the cut version of result of anomaly detection using the HTM algorithm. It detected 7 anomalies and 11 likely anomalies of high and medium levels, respectively. The detailed information on the detected anomalies is shown in Table 3. As noted in section 3, Table 3 will be used by the false-alarm labeling module as a reference in determining false alarms. It will find and label an alarm as false if the sensor data matches the date shown in the timestamp column.

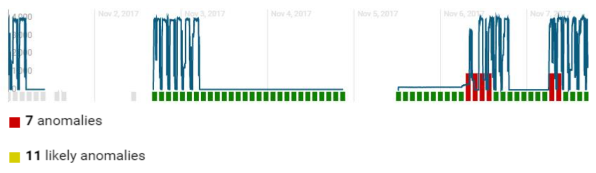


Fig. 7. Result of anomaly detection using HTM Algorithm (cut version)

In the final result, there are 7,287 sensor data labeled as false alarms, which accounts for 10% of the total sensor data.

IV. Conclusions

Based on the experimental results, we can conclude that our proposed method can successfully detect false alarms and label sensor data. A low rate of successful labeling of sensor data as false alarms is influenced by many factors, such as lack of consideration of the state of the machine (e.g. start state, stable state, shutdown state, etc.) by the process of sensor data logging, the result being that the sensor values are too dispersed. In the perspective of effectiveness, the proposed method will increase the efficiency of sensor data labeling. Redefining of the threshold value in each module is believed to increase the effectiveness of the automatic labeling of false alarms. To increase the validity of the false-alarm assessment, the results obtained from the present experimentation still need to be validated by a domain expert. The results presented here are limited to offline sensor data labeling, and the characterization of the anomalous merely based on one sensor data, not multiple sensor data. However, this method is useful as a baseline for further development of an automatic false-alarm detection to increase the safety-operation in the industry.

Today, in the real industry emerge the need to detect the false-alarm in a real-time manner. Therefore, further development of streaming analytics to facilitate real-time false-alarm detection and labeling as well as consideration of multi-sensor data for anomaly detection are two potential directions of future research.

REFERENCES

[1] ISA, ANSI/ISA-18.2: Management of Alarm Systems for the Process Industries. International Society of Automation. Durham, NC, USA, 2009.

[2] Nochur, A., Vedam, H., & Koene, J., Alarm performance metrics. Singapore Honeywell Singapore. 2001.

[3] Jain, P., Pasman, H. J., Waldram, S. P., Rogers, W. J., & Mannan, M. S., Did we learn about risk control since Seveso? Yes, we surely did, but is it enough? An historical brief and problem analysis. Journal of Loss Prevention in the Process Industries. 2016.

[4] Stauffer, T., Sands, N., & Dunn, D., Get a Life (cycle)! Connecting Alarm Management and Safety Instrumented Systems. Paper presented at the ISA Safety & Security Symposium. 2010b.

[5] Srinivasan, R., Liu, J., Lim, K., Tan, K., & Ho, W., Intelligent alarm management in a petroleum refinery. Hydrocarbon Processing, 83(11), 47-54. 2004.

[6] EEMUA. 191-Alarm Systems: A Guide to Design, Management and Procurement Edition 3. 2013.

[7] Pariyani, A., Seider, W. D., Oktem, U. G., & Soroush, M., Incidents Investigation and Dynamic Analysis of Large Alarm Databases in Chemical Plants: A Fluidized-Catalytic-Cracking Unit Case Study†. Industrial & Engineering Chemistry Research, 49(17), 8062-8079. 2010.

[8] S. Haque and S. Aziz, "False Alarm Detection in Cyber-physical Systems for Healthcare Applications", AASRI Procedia, vol. 5, pp. 54-61, 2013.

[9] S. Festag, "False alarm ratio of fire detection and fire alarm systems in Germany – A meta analysis", Fire Safety Journal, vol. 79, pp. 119-126, 2016.

[10] R. Pokrywka, "Reducing False Alarm Rate in Anomaly Detection with Layered Filtering", Computational Science – ICCS 2008, pp. 396-404, 2008.

[11] A. Patcha, J.M. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, Comput. Netw. 51 (12) (2007) 3448–3470.

[12] J. Lin, E. Keogh, A. Fu, H.V. Herle, Approximations to magic: finding unusual medical time series, in: 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05), IEEE, 2005.

[13] J. Wu, W. Zeng and F. Yan, "Hierarchical Temporal Memory method for time-series-based anomaly detection", 2018.

[14] S. Ahmad, A. Lavin, S. Purdy and Z. Agha, "Unsupervised real-time anomaly detection for streaming data", Neurocomputing, vol. 262, pp. 134-147, 2017.

[15] D. Rothenberg, Alarm Management for Process Control. New York: Momentum Press, 2011.

[16] J. Wang, F. Yang, T. Chen and S. Shah, "An Overview of Industrial Alarm Systems: Main Causes for Alarm Overloading, Research Status, and Open Problems", IEEE Transactions on Automation Science and Engineering, vol. 13, no. 2, pp. 1045-1061, 2016. Available: 10.1109/tase.2015.2464234.

[17] T. Adi et al., "Cloud-Based Predictive Maintenance Framework for Sensor Data Analytics", ICIC Express Letters, Part B: Applications, vol. 9, no. 11, p. 1161, 2018.

[18] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput.Surv. (CSUR) 41 (3) (2009) 15.

[19] V. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. 22 (2) (2004) 85–126.

[20] J. Hawkins, S. Ahmad, Why neurons have thousands of synapses, a theory of sequence memory in neocortex, Front. Neural Circ. 10 (2016).

[21] J. Hawkins, "Hierarchical Temporal Memory White Paper", Numenta, 2011.

[22] Jenkins, S., Guidelines for Engineering Design for Process Safety. Chemical Engineering,119(9), 9-10, 2012.

[23] Stauffer, T., & Clarke, P. Using alarms as a layer of protection. Process Safety Progress. 2015.

[24] Nochur, A., Vedam, H., & Koene, J., Alarm performance metrics. Singapore Honeywell Singapore. 2001.

[25] P. Goel, A. Datta and M. Mannan, "Industrial alarm systems: Challenges and opportunities", Journal of Loss Prevention in the Process Industries, vol. 50, pp. 23-36, 2017.

[26] S. Ahmad and S. Purdy, "Real-Time Anomaly Detection for Streaming Analytics", arXiv:1607.02480, 2016.

[27] Cui, Yuwei, Surpur, Chetan, Ahmad, Subutai, and Hawkins, Jeff. Continuous online sequence learning with an unsupervised neural network model. pp. arXiv:1512.05463 [cs.NE], 2015.

[28] Numenta GitHub repository. [Online]. Available: https://github.com/numenta/numenta-apps/tree/master/unicorn.

Authors

Taufik Nur Adi received the B.S degree in Computer Science from Sepuluh Nopember Institute of Technology, Indonesia in 2007. He received the M.S degree in Computer Science Bandung Institute of Technology, Indonesia, in 2012. He is interested in machine learning and reinforcement learning.

Hyerim Bae received Ph.D. degree in Industrial Engineering from Seoul National University, South Korea in 2002. He is currently a full professor of Business Process Management in the Department of Industrial Engineering, Pusan National University, Busan, South Korea. His research interests include business process management, business intelligence, software development, information technology, cloud computing, and logistics.

Nur Ahmad Wahid received the B.S degree in Computer Science from Sepuluh Nopember Institute of Technology, Indonesia in 2014. He is currently enrolled as a research student in Business Service Computing Laboratory Department of Industrial Engineering, Pusan National University, Busan, South Korea. His research interests include Machine Learning, Process Mining, and Business Process Simulation.