A Novel Framework for Context-aware Outlier Detection in Big Data Streams

Hussien Ahmad, Salah Dowaji
Damascus University
Syrian Arab Republic
hussien824@gmail.com
sdowaji@gmail.com

JDIM
Journal of Digital Information Management

ABSTRACT: *Outlier and anomaly detection has always been a critical problem in many fields. Although it has been investigated deeply in data mining, the problem has become more difficult and critical in the Big Data era since the volume, velocity and variety of data change drastically with rather complicated types of outliers. In such an environment, where real-time outlier detection and analysis over data streams is a necessity, the existing solutions are no longer effective and sufficient. While many existing algorithms and approaches consider the content of the data stream, there are few approaches which consider the context and conditions in which the content has been produced. In this paper, we propose a novel framework for contextual outlier detection in big data streams which inject the contextual attributes in the stream content as a primary input for outlier detection rather than using the stream content alone or applying the contextual detection on content anomalies only. The detection algorithm incorporates two approaches; the first, a supervised detection method and the other, an unsupervised, which allows the detection process to adapt to the normal change in the stream behavior over time. The detected outliers are either both content and contextual outliers or contextual outliers only. The proposed contextual detection approach prunes the false positive outliers and detects the true negative outliers at the same time. Moreover, in this framework, the detection engine preserves both outliers and context values in which those outliers were detected to be used in the engine self-training and in outliers modeling in order to enhance the outlier prediction accuracy.*

Subject Categories and Descriptors
H.2 [Database Management] H.2.8 Database Applications]; Data mining

General Terms:
Data Mining, Anomoly Detection, Contextual Detection

Keywords: Outlier Detection, Anomaly Detection, Context-Aware Outlier, Outliers Modeling, Big Data Analytics, Big Data Stream

Received: 17 April 2018, Revised 3 June 2018, Accepted 10 June 2018

DOI: 10.6025/jdim/2018/16/5/213-222

1. Introduction

Outlier detection and analysis is an important data mining problem that aims to find anomaly points and behavior in data sets (Zhang, 2008). This problem has been investigated deeply in a broad set of disciplines such as sensor networks, network intrusion, web logs, medical diagnosis and banking and insurance industries. The importance of outlier detection is imputed to the fact that the anomaly behavior is interpreted into important or critical information, for example, an anomaly behavior in computer network traffic might mean that there is an intrusion and that sensitive data is being sent to unauthorized destination (Dokas et al., 2002), or an abnormal MRI might indicate the existence of tumors (Seo & Milanfar, 2009).

The outlier detection problem has many aspects related to input data, types of anomalies, detection algorithms and detection process output. In general, outliers are categorized into three types (Chandola et al., 2007): (i) point outliers: a data point is considered as outlier when viewed against the whole dataset, (ii) contextual outliers: a data point is considered outlier when viewed against contextual information linked to the dataset and (iii) collective outliers: a data point is considered outlier when viewed collectively with other points against the whole dataset. The detection algorithms can also be categorized into three types: supervised, semi-supervised and unsupervised (Chandola et al., 2007), this classification depends on the availability of training datasets, with or without labeling of entries as normal and abnormal. The output of the outliers' detection process might be assigning an anomaly score to each data point or simply a threshold-based decision on the data point to be outlier or not (Chandola et al., 2007).

The traditional outlier detection methods process only the content of the dataset and based on the content assigns anomaly score for each data point, while the contextual outlier detection looks into the influence of external factors on the dataset and process both the content of the dataset and its context at the same time. The advantage of integrating contextual factors in the detection process is not limited to minimizing the false alarms or pruning the false positive outliers but exceeds that to detect the hidden outliers which act as normal data point while they are real outliers if looking at the context they were generated in. As an example for false alarm, a spike in temperature reading might be taken as anomalous for a sensor, independent of external factors, but at that particular time of the year, spikes in temperature might be a possibility and can be considered as a normal reading. On the other hand, as an example for hidden outliers, a moderate reading of temperature might be considered normal, independent from external factors but considering the time of the year, it might be an outlier.

Although the outlier detection has been investigated in different fields of research, the problem has become more complicated in the Big Data era. The requirements of real time and parallel processing that the Big Data environments require in addition to the massive volume of data and high velocity, variety and veracity make it necessary and critical to investigate this field again (Rettig et al., 2015), especially because many outlier detection algorithms such as k-nearest neighbors, support vector machine and cluster analysis were designed to run on single machine and usually work with relatively small and static datasets.

The research described in this paper introduces a novel framework for contextual outlier detection in Big Data streams. The design of the framework depends on Big Data tools Kafka which consumes and prepares the main data stream and any other external contextual sources as micro-batches which will be delivered to Spark where the detection algorithms are applied. The detection method is a mix of supervised and unsupervised methods which enables the detection of outliers, minimizing the false alarms and hidden outliers and at the same time adapts to the normal change in the data stream behavior. The nature of the detection method enables the identification of outliers which might have influence on the future behavior of the data stream and can enhance the prediction capabilities of the data stream behavior. The proposed framework is designed to work in different settings and to handle any type of data stream. Also, this framework is novel in its design to handle Big Data streams and its capabilities to detect hidden outliers and minimizing the false alarms. Furthermore, the framework is designed to integrate semantic analysis engines which enables the detection over text data and also enhance the quality of the detection process.

The sections of this paper are organized as follows: The "Related work" section reviews relevant work, conducted in the field of outlier detection in data streams. The "Framework Overview" describes the general architecture of the proposed framework and the "Outlier detection method" discusses the details of the detection algorithm. Experiments conducted and their results are presented in "Experiments and discussion". Finally, the paper concludes with conclusions and recommendations for future work.

2. Related Work

Outlier detection problem has its applications in a variety of fields. The detection algorithms can be categorized into point detection, collective detection and contextual detection (Chandola et al., 2009). This paper focuses on contextual detection algorithms. Contextual outlier detection in data streams aims at finding anomalous and abnormal data points over the data stream taking into consideration multiple context parameters which might be internal or external. Although the research focuses on big data streams settings while addressing the outlier detection problem, we review some contextual outlier detection in data streams which do not consider the big data settings since the outlier detection methods in big data settings are still limited.

Although there are several algorithms which process the contextual parameters to detect the outliers over data stream, those algorithms were designed according to specific applications requirements. Hayes and Capretz propose a framework for outlier detection in big sensor data streams (Hayes & Capretz, 2015), the framework consists of two main components: content detector and context detector. The primary reason for separating the main components is to enable scalability for huge volumes of data. The content detector handles all coming data points and detect point outlier using univariate Gaussian predictor based on a predefined training dataset while the context detector examines only the outliers detected in the first stage to prune the false-positive outliers using a

multi-variate Gaussian predictor. The main advantage of this framework is its scalability to process high volumes of data while minimizing the false alarms. However, this framework does not adapt to normal evolution of the data stream nor detect the hidden outliers which behave as normal data points when processing the content of the data stream despite the context parameters. Furthermore, this framework was designed to handle sensor data streams and works in a predefined environment.

Jiang et al. propose an unsupervised method for collective contextual outlier detection over multiple data streams (Jiang et al., 2014). This method calculates the anomaly score for the current snapshot and uses the score to calculate the stream anomaly score based on the amount of entropy that each data point brings to the current snapshot. It also uses a historical decay factor in order to include the historical behavior of each data stream in the current snapshot processing. The triggering engine adopts the anomaly score of each data stream to decide whether a specific stream is anomalous or not, in the current snapshot, based on the dynamic threshold that is calculated with the historical anomaly score of each data stream. The proposed method enables the detection of transient fluctuation which minimizes the false alarms. The advantage of this method comes from the fact that it can adapt to the normal evolution behavior and using dynamic threshold to detect outliers. Also, this method is designed to detect outliers in real time and can handle massive volumes of data. On the other hand, the method does not clarify how to include contextual parameters in the detection process and considers the different data streams to be independent from each other without any correlation.

Mahapatra et al. propose an approach for outlier detection in large corpus of documents and to minimize the false alarms based on the semantic content of the context parameters (Mahapatra et al., 2012). This approach depends on the Latent Dirichlet Allocation (LDA) algorithm for statistical analysis of the documents' content and detection of anomalous topics. Afterwards, the method uses semantic analysis algorithms to analyze the context of each topic using WordNet and Normalized Google Distance. This approach uses semantic analysis algorithms to detect contextual outliers, however, it depends on a predefined threshold and runs in predefined conditions and static datasets.

AlEroud and Karabatis propose contextual anomaly detection approach to discover zero-day cyber-attacks (AlEroud & Karabatis, 2012). This approach consists of two main parts; content detection and contextual detection with small modules for preprocessing, sampling and profiling. The contextual misuse module uses a conditional entropy-based technique to create known attacks context profiles based on historical and training data. It calculates a profile similarity score; in case of a complete match with one of the known attacks, the record is announced as anomalous while in case of high partial matching, the record might imply a new attack type or zero-day attack. In order to detect such attacks, the anomaly detection module implements one class nearest neighbor algorithm with Euclidian distance measure. This approach uses predefined context parameters and predefined threshold for the 1-NN algorithm, also the conditional entropy-based calculation is complex.

While there are few works handling the problem that this research paper focuses on, there are still many techniques and approaches have already been worked on, for outlier detection in data streams. Gupta et al. has categorized these techniques into (Gupta et al., 2014): (i) evolving prediction models like using online discounting models (Yamanishi & Takeuchi, 2002) (Aggarwal, 2005), dynamic cluster maintenance (Sreevidya, 2015) (Sequeira & Zaki, 2002) (Aggarwal & Philip, 2010) and dynamic Bayesian networks (Hill et al., 2007), (ii) distance-based outliers for sliding window detecting both global outliers (Angiulli & Fassetti, 2007) (Yang et al., 2009) (Bu et al., 2009) (Cao et al., 2010) and local outliers (Breunig et al., 2000) (Pokrajac et al., 2007), outlier in high dimensional data streams (Zhang et al., 2009) and outliers in distributed data streams using three different models: sharing local outliers and data points (Branch et al., 2013), sharing local outliers only (Zheng et al., 2015) (Otey et al., 2006) and sharing local outliers and data distributions (Subramaniam et al., 2006) (Palpanas et al., 2003). These techniques handle the outlier detection problem in silos, mostly focusing on the content of the stream while the focus in this research is to provide the holistic approach for contextual outlier detection in Big Data streams.

More comprehensive surveys on outlier detection problem in different settings are proposed by: Gupta et al. (2014) for outlier detection in temporal data; Akoglu et al. (2015) for graph based anomaly detection; Zhang et al. (2010) for outlier detection in wireless sensor networks; and Zhang (2008) for point outlier detection from vector like datasets.

3. Framework Overview

This section provides an overview of the contextual outlier detection framework that is designed as a holistic solution for outlier detection in Big Data streams. The main features that the framework is trying to provide can be summarized as:

- Detection of all outliers (relevant and correct semantically and contextually).
- Minimizing the false alarms.
- Enabling outlier storage for future analysis.
- Enabling outliers' analysis to enhance the detection prediction process and
- Enabling human interaction with the detection process.

In order to provide the above features, a conceptual framework (Figure 1) has been proposed as a guideline

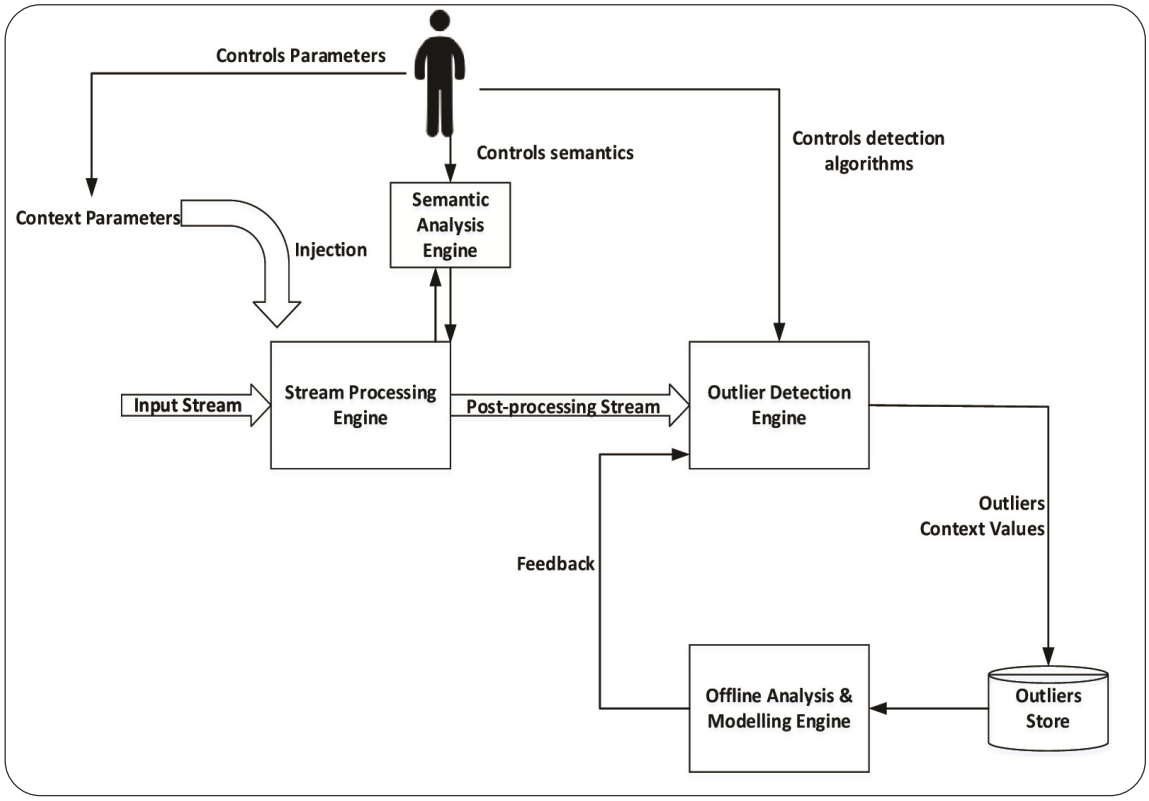

Figure 1. Conceptual framework for outlier detection in Big Data streams

for any implementation and case study. The stream processing engine and outlier detection engine form the core modules of the framework while the semantic analysis engine and outlier modeling and analysis engine are optional and complementary.

The implementation of this conceptual framework has to consider Big Data settings in addition to support different types of input streams and enabling external contextual parameters. Following is the description of all modules.

• **Stream Processing Engine:** This engine is implemented using Kafka instance with a custom Kafka producer which integrate the main stream input with external contextual parameters provided by the external human analyst based on the timestamp of each coming data point. The output of the engine is a matrix of normalized real numbers. In case the type of main stream or any of the contextual parameters are not numeric, then the Semantic Analysis Engine is used to convert the values into normalized real numbers. The output matrix forms a micro-batch that will be delivered to the Outlier Detection Engine through a Kafka topic. The selection of Kafka is due to its flexibility of incorporating multiple data sources and scalability to handle high-volume data streams and the ability to implement the Map/Reduce programming model.

• **Outlier Detection Engine:** This engine is implemented using Spark Streaming instance which handles micro-batches received through Kafka topic. The core detection functionality is implemented in this engine. The engine processes all coming data points and applies contextual detection algorithm on every single batch which guarantees the detection of all hidden outliers and pruning of false alarms. The output of this engine is the detected outliers which will be stored in Outlier Store with their contextual parameters. The selection of Spark is due to its streaming processing model enabling the processing of micro-batches which fits with the problem under consideration; in addition to its scalability features.

• **Semantic Analysis Engine:** The implementation of this engine is dependent on the type of the case under investigation. For example, when dealing with a Twitter public stream, the implementation of this engine should provide semantic analysis and similarity functions using WordNet (Pedersen, et al., 2004) and simultaneously, sentiment analysis using Stanford NLP sentiment analysis engine (Manning et al., 2014). The implantation of the Semantic Analysis Engine is beyond the scope of this paper and will be presented and tested in a separate work, using Twitter public stream and other datasets for testing.

• **Outlier Storage:** The Outlier Detection Engine stores all detected outliers along with their context parameters and timestamp in a persistent storage (CSV files for instance) which should accessible for the Analysis Engine. The minimum information that the storage engine should store are the data point, the context parameter, the outlier type and the timestamp. The stored information can be extended to store all data points and their context in the current window to enable a holistic view on the outliers and the context under which they have been generated.

The storage process starts after getting the output of the outlier detection engine for the current window in order to reduce the processing time of the window inside the detection engine and to enable the storage of all data points of the current window if any outliers have been detected.

• **Offline Analysis & Modelling Engine:** This is an offline engine and can be implemented in different ways, depending on the nature of the problem and preference of the data analyst.

The proposed design and implementation of the framework guarantee scalability to massive volume of data, especially because the detection algorithm can easily be run in distributed settings using the Map/Reduce model. Furthermore, the separation between processing module and detection module enhance performance since the processing module prepares the micro-batch "Matrix of Real Numbers" while the detection module processes the previous batch.

The implementation of this framework can be extended to provide different outliers detection algorithms and semantic analysis functionality. During execution time, the data analyst can choose which algorithm is most suitable to the case under study.

4. Outlier Detection Method

4.1. Data Receiving and Preparation

The role of this stage is to collect data from different data sources (the main stream and the external contextual parameters) and then perform the injection process to inject all inputs into one single stream. In case there are text parameters, the semantic analysis engine will be called at this stage in order to transform the text data into real numbers. All data points received during the same timeframe will be gathered together in an array of real numbers which will be delivered to the detection stage as one window. The advantage of separating the data preparation stage from outlier detection is that the preparation engine is able to perform some time consuming tasks before it delivers data to detection engine while the detection engine will be processing the previous window. The length of the sliding window is configured based on each problem and system capabilities. The sliding windows in this configuration is not overlapping and thus each data point will be processed only once as it belongs to a single sliding window.

When the problem scales up to a high number of external contextual parameters and high speed data streams, the preparation stage can easily be distributed over multiple processing nodes using the Map/Reduce programming model where the output of the reduce stage will be the final array of real numbers forming the sliding window, which will be delivered to the detection stage.

4.2. Anomaly Scoring Stage

The anomaly scoring stage is the core stage of the framework. This stage consists of two sub-stages: the supervised scoring and the unsupervised scoring. The output of this stage is a tuple of Boolean for each data point indicating whether the point is anomalous based on the supervised and unsupervised scoring.

We consider that the data stream follows a normal distribution and thus the statistical properties of normal distributions will be applied in the scoring stage.

4.2.1. Supervised Anomaly Scoring

The supervised anomaly scoring stage depends on a predefined training dataset and uses multivariate Gaussian algorithm to calculate the anomaly score for each data point in the current window. The multivariate Gaussian algorithm depends on the mean vector and the covariance matrix of the training dataset.

Definition (1): Supervised anomaly score is the value of the probability density function of the multivariate normal distribution of the training dataset.

A data point $X(x1, x2, \ldots, xk)$ is considered anomalous if its anomaly score is less than $\in$, which is the threshold that is defined based on the multivariate normal distribution of the training dataset.

The calculation of anomaly score using the normal distribution of the training dataset will not be able to adapt to the normal evolution of the data stream and thus the detection process might result in a lot of false-positive and hidden outliers at the same time, due to the transient fluctuation and phase shift of the data stream. In order to avoid such situations, a dynamic normal distribution is being calculated using the historical evolution of the data stream. The historical factor is calculated based on the current window mean vector and covariance matrix using a factor ($\lambda > 0$) that controls the decay speed.

$$\mu_{new} = \mu_{old} + e^{-\lambda}(\mu_{old} - \mu_{current\_window})$$

$$COV_{new} = COV_{old} + e^{-\lambda}(COV_{old} - COV_{current\_window})$$

The new mean vector and covariance matrix will be used to calculate the anomaly score for the data points in the next sliding window.

The output of this sub-stage is a vector of Boolean values indicating whether each data point in the window is anomalous or not based on the supervised anomaly score.

4.2.2. Unsupervised Anomaly Scoring

The unsupervised anomaly score is used to determine the local outliers in the current sliding window. We adapt the method used by Jiang et al. (Jiang et al., 2014)

***Definition (2)*:** Unsupervised anomaly score is the amount of uncertainty the data point brings to the window.

Since the data stream follows normal distribution, it would be efficient to use the amount of entropy each data point brings to the window to calculate the anomaly score. The entropy of normal distribution is given by the equation:

$$Ent = \frac{1}{2} In(2\pi\sigma^2)$$

The amount of entropy that a data point X brings to the window is the difference between the entropy of normal distribution in presence of X and in absence of X. This calculation requires two types of variance: variance and leave-one-out variance of the distribution when X is not counted.

In order to reduce the complexity of the leave-one-out variance calculation, the distance matrix M is used to store the distance between the value of each dimension and the corresponding mean. Thus, the leave-one-out variance can be calculated quickly using the following equation:

$$\sigma^2_{ik} = \frac{n\sigma^2_k - M^2_{i,k}}{n-1}$$

Where σ^2_{ik} is the leave-one-out variance excluding the point k from the window, σ^2_k is the variance of the window and "i" is the dimension. Thus, the amount of entropy that a data point x brings to the window can be calculated using the following equation:

$$N_x = Ent(x) = \sum_{i=1}^{m} In \frac{n(sum(M_i) - M_{i,x})}{(n-1)(sum(M_i))}$$

After calculation of the unsupervised anomaly score for each point in the window, unsupervised method is used to determine whether a data point X is an anomaly or not. This method requires the median and the minimum of the anomaly scores of the data point in the window. A data point X is considered anomaly if

$$N_x > 2\ (N_{median} - N_{min})$$

The output of this sub-stage is a vector of Boolean values indicating whether each data point in the window is anomalous or not, based on unsupervised anomaly score.

4.3. Alert Triggering

Considering the two methods of anomaly scoring, the output of the anomaly scoring stage will be two Boolean vectors indicating whether each point is an anomaly based on the supervised and unsupervised scoring. The output Boolean vectors will be of the same size of the input matrix and the index of each entry will be used to determine the specific data point. The possible output for each data point is shown in the following fact table (Table 1):

The outlier storage engine will store all outlier points preserving the point itself, the contextual parameters and the type of outlier based on the supervised and unsupervised scoring. The storage of this information about the outliers will enable further analysis of the outliers and the conditions under which they have been generated and also enables better prediction of future outliers.

Supervised Scoring	Unsupervised Scoring	Result
True	True	General Outlier
True	False	General Outlier
False	True	Local Outlier
False	False	Normal

Table 1. Fact table of output of anomaly scoring stage

The outlier storage engine can be implemented based on the specific nature of the problem under investigation and should provide a persistent data store. In the current case and for test purpose the storage is done using CSV files that can be used with different analysis tools and loaded into any type of databases.

4.4. Output Analysis

The proposed detection method has two main components: (i) the dynamic supervised anomaly scoring using multivariate Gaussian algorithm that guarantees the inclusion of contextual parameters in the detection process for each point which in turn prunes the false positive alarms and detects the hidden outliers; (ii) the unsupervised anomaly scoring using the amount of entropy each data point brings to the current window which indicates the position of the current window in the general evolution of the data stream.

The output of the detection process has four possible values as shown above:

• **False – False:** This value means that the data point is normal in the data stream.

• **True – True:** This value means that the data point is an outlier located far from the other points in the current window and also from other points in the previous windows.

• **True – False:** This value means that the data point is a general outlier located far from points in the previous windows of the data stream while it is still located close to other points in the current window; this indicates that most points in the current window might be of this type and the window itself is behaving as an outlying window.

• **False – True:** This value means that the data point is a local outlier located close to the points in the previous windows while it is located far from other points in the current window; this indicates that most points in the current window might be of this type and the window itself is behaving as an outlying window.

Looking closely to the (True – False and False – True)

possible results, we notice that these two values will come together in one window. The anomaly behaviour of a window indicates for either severe fluctuation in the data stream or a start of a major change in the data stream behaviour. Furthermore, the data points giving such anomaly values are still part of the general system of the data stream even if they do not fall in any of the normal data points classes; such points are random and have effect on the future of the data stream.

In order to predict whether an outlier window is a start of the behavioural change or just a temporal fluctuation in the data stream, the Offline Analysis and Modelling Engine is used to build classes out of the detected outliers based on the context parameters for each outlier to check whether a new outlier window falls into one of these classes or not. If a series of consequent windows were detected as outliers, then the first one would be considered as a start of a major change in the behaviour of the data stream while isolated outlier window would indicate a severe fluctuation in the data stream.

Since the detection method employs a dynamic normal distribution using the evolution of the data stream, it would be suitable to adopt the presented analysis with a high level of confidence.

5. Experiments and Discussion

5.1. Experiment Settings

The proposed framework was implemented using Kafka 2.11 and Spark 2.3.0. The code has been written in Java and compiled using jdk1.8.0. All tests were run on 2.30GHz core I7, 12G RAM running Windows 10. The tests were run in a single processing node configuration for both Spark and Kafka.

Several tests were conducted using predefined datasets from UCI machine learning repository. Two tests used the Air Quality dataset: The first one used the attribute "C6H6(GT)" (True hourly averaged Benzene concentration in microg/m^3) as primary content of the data stream and the attribute "*T*" (Temperature in °C) as a context parameter. The second test used the attribute *"RH"* (Relative Humidity (%)) as primary content of the data stream and the attribute "*T*" (Temperature in °C) as a context parameter. The Air Quality dataset contains 9357 instances covering the period 10th Match 2004 to 4th April 2005. For the two tests we used data of March, April, May and June as training dataset (1974 instances) and the rest as test data (7383 instances). The high Pearson correlation between "*T*" and "C6H6(GT)" is 0.97 and between "*T*" and "*RH*" is 0.93 which indicate high level of dependency and correlation between the selected attributes and form a good choice for content and context parameters of the test data streams.

The dataset itself contained global, local and hidden outliers while it did not include false alarms. In order to cover all types of outliers we injected four data points into the test dataset of type false alarms.

The primary target of these injections is to test whether the proposed detection method is able to detect specific types of outliers: real outliers, false alarms, hidden outliers using the two anomaly scoring methods. The dataset itself contained real outliers and hidden outliers and we injected some false alarms.

In order to demonstrate the concept of contextual outliers, we started the experiments by running the test against the content attribute regardless the context. The results showed that the algorithm was able to detect all real outliers in addition to the false alarms. The second phase of testing was to run the test against the content attribute taking the context into consideration; the results showed that the algorithm was able to detect all real outliers, hidden outliers and pruned the false alarms.

5.2. Results Analysis

In this section we describe and analyse the results of the experiment on "C6H6(GT)" attribute as primary content of the data stream and "*T*" as context attribute.

During the first phase of the experiment, the algorithm was able to detect 329 outliers including the 4 false alarms but was not able to detect the 20 hidden outliers using the supervised anomaly scoring and 50 outliers using the unsupervised anomaly scoring. While the results of the second phase with context attribute included in the test, the algorithm detected 345 outliers including the 20 hidden outliers and pruning the 4 false alarms using the supervised anomaly scoring and 55 outliers using the unsupervised anomaly scoring.

The detected outliers were of types: True-True (0 outliers), True-False (345), False-True (55).

Most of the detected outliers were of the type True-False (global outliers) due to the design of the dataset. The method was able to detect 20 hidden outliers as a result of using the context parameter in the detection process; those outliers were not detected using the content detector alone.

The 55 local outliers were distributed over the full dataset without any concentration; the distribution of the local outliers indicates normal evolution of the data stream and the local outliers were the at the early stage of the change in the stream behavior. The Figure 2 shows the global and local outliers.

The tests were run using a high value of the decay factor in order to keep high influence of the training dataset during the detection process. The value of the decay factor indicates the confidence the analyst has on the training dataset and on the evolution of the data stream. Figure 3 illustrates the evolution of the window average for both attributes (C6H6(GT) and Temperature) during the test starting from the values given by the training dataset.

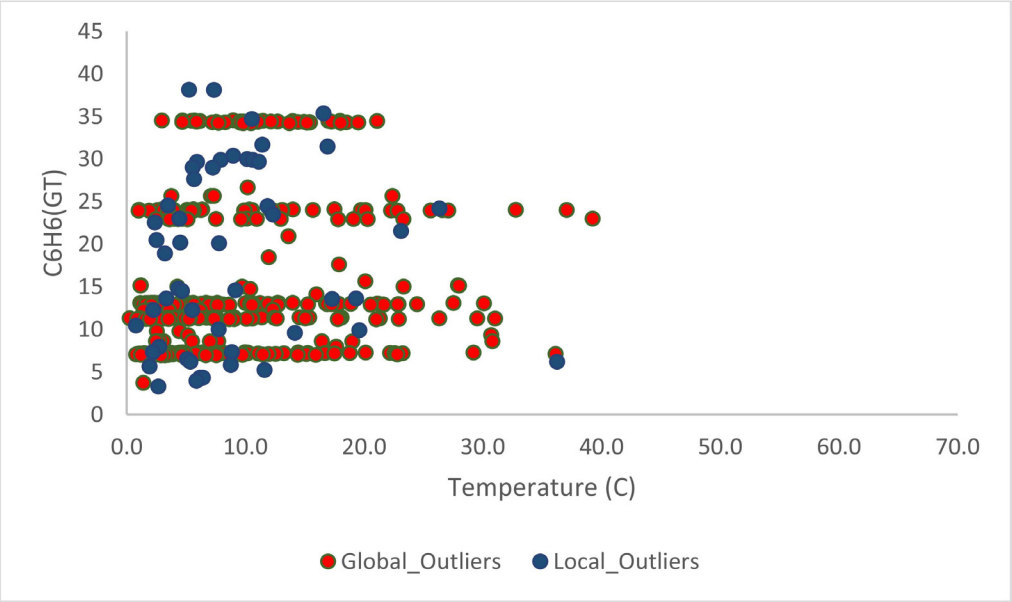

Figure 2. Detected global and local outliers

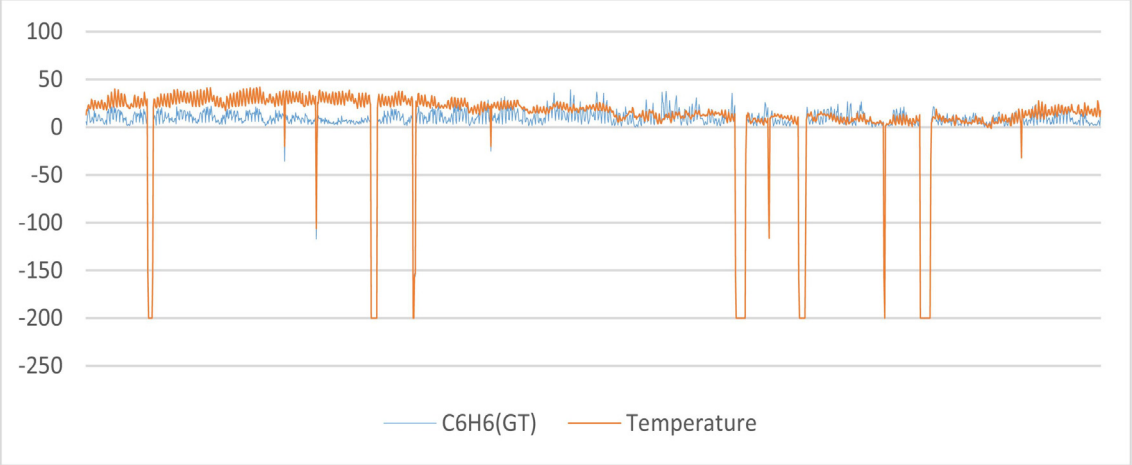

Figure 3. Window average evolution during the test

The experiment was run under conditions in which data points were generated at high speed. The difference in timestamp between data points was set to 0.1 milliseconds. The time needed to process a window of 10 data points was approximately 0.12 milliseconds.

The proposed framework and detection algorithm showed high level of efficiency and accuracy in detecting all types of outliers and pruning false alarms.

6. Conclusion

In this paper, we propose a novel framework for outlier detection in Big Data streams to detect anomalies based on the content and the context of the data stream. The proposed outlier detection method employs supervised and unsupervised detection techniques at the same time. The supervised detection identifies global outliers along the data stream based on dynamic training datasets that evolves and adapts according to the behaviour of the data stream; while the unsupervised detection identifies the local outliers in the current sliding window. The combination of supervised and unsupervised detection techniques enabled the detection process to adapt to the normal behaviour change of the data stream and also to enhance the predictability of changes at early stages. The framework, provides also the possibility to preserve detected outliers for further offline analysis and modelling. To demonstrate the effectiveness and efficiency of the proposed framework, it was implemented using Big Data tools Spark and Kafka and tested using several real and predefined datasets.

In future, we plan to enhance the semantic analysis engine and also to provide the possibility of online interaction with the detection process. Another improvement to the framework is to implement it in a distributed model using the map/reduce programming model for both the

preparation and detection modules.

Furthermore, on the theoretical part related to the type of detected outliers, we would like to bring more concrete examples and tests to prove the effect of the random data on the future behaviour of the data stream.

References

[1] Aggarwal, C. C. (2005). On abnormality detection in spuriously populated data streams. *In*: Proceedings of the 2005 SIAM International Conference on Data mining (p. 80-91). Society for Industrial and Applied Mathematics. (April).

[2] Aggarwal, C. C., Philip, S. Y. (2010). On clustering massive text and categorical data streams. *Knowledge and Information Systems*, 24 (2) 171-196.

[3] Akoglu, L., Tong, H., Koutra, D. (2015). Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29 (3) 626-688.

[4] AlEroud, A., Karabatis, G. (2012). A contextual anomaly detection approach to discover zero-day attacks. In Cyber Security (CyberSecurity), 2012 International Conference on (p. 40-45). IEEE. (December).

[5] Angiulli, F., Fassetti, F. (2007). Detecting distance-based outliers in streams of data. *In*: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (p. 811-820). ACM. (November).

[6] Apache Kafka, https://kafka.apache.org/

[7] Apache Spark, https://spark.apache.org/

[8] Branch, J. W., Giannella, C., Szymanski, B., Wolff, R., Kargupta, H. (2013). In-network outlier detection in wireless sensor networks. *Knowledge and Information Systems*, 34 (1) 23-54.

[9] Breunig, M. M., Kriegel, H. P., Ng, R. T., Sander, J. (2000). LOF: identifying density-based local outliers. *ACM sigmod record* 29 (2) 93-104. ACM. (May).

[10] Bu, Y., Chen, L., Fu, A. W. C., Liu, D. (2009). Efficient anomaly monitoring over moving object trajectory streams. *In*: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (p. 159-168). ACM. (June).

[11] Cao, H., Zhou, Y., Shou, L., Chen, G. (2010). Attribute outlier detection over data streams. In International Conference on Database Systems for Advanced Applications (pp. 216-230). Springer, Berlin, Heidelberg. (April).

[12] Chandola, V., Banerjee, A., Kumar, V. (2007). Outlier detection: A survey. ACM Computing Surveys.

[13] Chandola, V., Banerjee, A., Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 15.

[14] Gupta, M., Gao, J., Aggarwal, C., Han, J. (2014). Outlier detection for temporal data. Synthesis Lectures on Data Mining and Knowledge Discovery, 5 (1) 1-129.

[15] Hae Jong Seo, Milanfar, P. (2009). A Non-Parametric Approach To Automatic Change Detection In MRI Images of The Brain. IEEE International Symposium on Biomedial Imaging: From Nano to Macro.

[16] Hayes, M. A., Capretz, M. A. (2015). Contextual anomaly detection framework for big sensor data. *Journal of Big Data*, 2 (1) 2.

[17] Hill, D. J., Minsker, B. S., Amir, E. (2007). Real-time Bayesian anomaly detection for environmental sensor data. *In*: Proceedings of the Congress-International Association for Hydraulic Research 32 (2) p. 503. (July).

[18] Jiang, Y., Zeng, C., Xu, J., Li, T. (2014). Real time contextual collective anomaly detection over multiple data streams. Proceedings of the ODD, 23-30.

[19] Mahapatra, A., Srivastava, N., Srivastava, J. (2012). Contextual anomaly detection in text data. *Algorithms,* 5 (4) 469-489.

[20] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. *In*: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations (p. 55-60).

[21] Otey, M. E., Ghoting, A., Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. Data mining and knowledge discovery, 12 (2-3), 203-228.

[22] Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D. (2003). Distributed deviation detection in sensor networks. *ACM SIGMOD Record*, 32 (4) 77-82.

[23] Paul Dokas, E Rtoz, L., K Umar, V., L Azarevic, A., S Rivastava, J., Ning Tan, P. (2002). Data mining for network intrusion detection. Proceedings of the NSF Workshop on Next Generation Data Mining.

[24] Pedersen, T., Patwardhan, S., Michelizzi, J. (2004). WordNet: Similarity: measuring the relatedness of concepts. *In:* Demonstration papers at HLT-NAACL 2004 (p. 38-41). Association for Computational Linguistics. (May).

[25] Pokrajac, D., Lazarevic, A., Latecki, L. J. (2007). Incremental local outlier detection for data streams. *In:* Computational intelligence and data mining, 2007. CIDM 2007. IEEE symposium on(p. 504-515). IEEE. (March).

[26] Rettig, L., Khayati, M., Cudré-Mauroux, P., Piórkowski, M. (2015). Online anomaly detection over big data streams. *In*: 2015 IEEE International Conference on Big Data (Big Data) (p. 1113-1122). IEEE. (October).

[27] Sequeira, K., Zaki, M. (2002). ADMIT: anomaly-based data mining for intrusions. *In*: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (p. 386-395). ACM. (July).

[28] Sreevidya, S. (2015). Detection of Outliers in Data

Stream Using Clustering Method. *International Journal of Science, Engineering and Technology Research* (IJSETR)/ 2015/2278-7798, 4.

[29] Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D. (2006). Online outlier detection in sensor data using non-parametric models. *In*: Proceedings of the 32nd international conference on Very large data bases (p. 187-198). VLDB Endowment. (September).

[30] Yamanishi, K., Takeuchi, J. I. (2002). A unifying framework for detecting outliers and change points from non-stationary time series data. *In*: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (p. 676-681). ACM. (July).

[31] Yang, D., Rundensteiner, E. A., Ward, M. O. (2009). (March). Neighbor-based pattern detection for windows over streaming data. *In*: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (p. 529-540). ACM.

[32] Zhang, J. (2008). Towards outlier detection for high-dimensional data streams using projected outlier analysis strategy (Doctoral dissertation, Dalhousie University Halifax).

[33] Zhang, J., Gao, Q., Wang, H., Liu, Q., Xu, K. (2009). Detecting projected outliers in high-dimensional data streams. In International Conference on Database and Expert Systems Applications (p. 629-644). Springer, Berlin, Heidelberg. (August).

[34] Zhang, Y., Meratnia, N., Havinga, P. (2010). Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials*, 12 (2) 159-170.

[35] Zheng, Z. G., Jeong, H. Y., Huang, T., Shu, J. (2015). KDE based outlier detection on distributed data streams in sensor network, *J Sens*, 2015, 1-11