6-2014

Contextual Anomaly Detection in Big Sensor Data

Michael Hayes
Western University, mahyes34@uwo.ca

Miriam A M Capretz
Western University, mcapretz@uwo.ca

Citation of this paper:

M. A. Hayes, M. A. M. Capretz, "Contextual Anomaly Detection in Big Sensor Data", in Proc. of the 3rd Int. Congress on Big Data (IEEE BigData 2014), June 27-July 2, 2014, Anchorage, Alaska, USA.

Contextual Anomaly Detection in Big Sensor Data

Michael A. Hayes, Miriam A.M. Capretz
Department of Electrical and Computer Engineering
Western University
London, Ontario, Canada N6A 5B9
{mhayes34, mcapretz}@uwo.ca

***Abstract*—Performing predictive modelling, such as anomaly detection, in Big Data is a difficult task. This problem is compounded as more and more sources of Big Data are generated from environmental sensors, logging applications, and the Internet of Things. Further, most current techniques for anomaly detection only consider the content of the data source, i.e. the data itself, without concern for the context of the data. As data becomes more complex it is increasingly important to bias anomaly detection techniques for the context, whether it is spatial, temporal, or semantic. The work proposed in this paper outlines a contextual anomaly detection technique for use in streaming sensor networks. The technique uses a well-defined content anomaly detection algorithm for real-time point anomaly detection. Additionally, we present a post-processing context-aware anomaly detection algorithm based on *sensor profiles*, which are groups of contextually similar sensors generated by a multivariate clustering algorithm. Our proposed research has been implemented and evaluated with real-world data provided by Powersmiths, located in Brampton, Ontario, Canada.**

***Keywords*-Big Data Analytics, Contextual Anomaly Detection, Predictive Modelling, Multivariate Clustering**

I. Introduction

Anomaly detection is the identification of abnormal events or patterns that do not conform to expected events or patterns [1]. Detecting anomalies are important in a wide range of disparate fields; such as, diagnosing medical problems, bank and insurance fraud, network intrusion, and object defects. Generally anomaly detection algorithms are designed based on one of three categories of learning: unsupervised, supervised, and semi-supervised [1]. These techniques range from training the detection algorithm using completely unlabelled data to having a pre-formed dataset with entries labelled *normal* or *abnormal*. A common output of these techniques is a trained categorical classifier which receives a new data entry as the input, and outputs a hypothesis for the data points abnormality. One problem with standard anomaly detection approaches is that there is little concern for the context of the data content. For example, a sensor reading may determine that a particular electrical box is consuming an abnormally high amount of energy. However, when viewed in context with the location of the sensor, current weather conditions, and time of year, it is well within normal bounds. These types of anomalies are commonly found in fields with spatial, sequential, or temporal attributes that can be associated with the sensor [1].

One interesting, and growing, field where anomaly detection is prevalent is in *Big sensor Data*. Sensor data that is streamed from sources such as electrical outlets, water pipes, telecommunications, Web logs, and many other areas, generally follows the template of large amounts of data that is input very frequently. In all these areas it is important to determine whether faults or defects occur. In electrical and water sensors, this is important to determine faulty sensors, or deliver alerts that an abnormal amount of water is being consumed, as an example. In Web logs, anomaly detection can be used to identify abnormal behavior, such as identify fraud. In any of these cases, one difficulty is coping with the velocity and volume of the data while still providing real-time support for detection of anomalies.

An area that has not been well explored in literature is contextual anomaly detection for sensor data. Some related works have focused on anomaly detection in data with spatial relationships [2], while others propose methods to define outliers based on relational class attributes [3]. A prevalent issue in these works is their scalability to large amounts of data. In most cases the algorithms have increased their complexity to overcome more naive methods, but in doing so have limited their application scope to offline detection. This problem is compounded as *Big Data* requirements are found not only in giant corporations such as Amazon or Google, but in more and more small companies that require storage, retrieval, and querying over very large scale systems. As Big Data requirements shift to the general public, it is important to ensure that the algorithms which worked well on small systems can scale out over distributed architectures, such as those found in cloud hosting providers. Where an algorithm may have excelled in its serial elision, it is now necessary to view the algorithm in parallel; using concepts such as divide and conquer, or MapReduce [4]. Many common anomaly detection algorithms such as k-nearest neighbour, single class support vector machines, and outlier-based cluster analysis are designed for single machines [5]. A main goal of this research is to leverage existing, well-defined, serial anomaly detection algorithms and redefine them for use in Big Data analytics.

The research in this paper will describe a technique to detect contextually anomalous values in streaming sensor systems. This research is based on the notion that anomalies have dimensional and contextual locality. That is, the dimensional locality will identify those abnormalities which are found to be structurally different based on the sensor reading. Contextually, however, the sensors may introduce new information which diminishes or enhances the abnormality of the anomaly. For example, sensors may produce what

M.A. Hayes, M.A.M. Capretz, Contextual Anomaly Detection in Big Sensor Data, The 3rd IEEE International Congress on Big Data, 2014.

appears to be anomalous readings at night for electrical sensor data; however, when introduced with context such as time of day, and building business hours, anomalous readings may be found to be false positives. To cope with the volume and velocity of Big Data, the technique will leverage a parallel computing model, MapReduce. Further, the technique will use a two-part detection scheme to ensure that point anomalies are detected in real-time and then evaluated using contextual clustering. The latter evaluation will be performed based on *sensor profiles* which are defined by identifying sensors that are used in similar contexts. The primary goal of this technique is to provide a scalable way to detect, classify, and interpret anomalies in sensor-based systems. This ensures that real-time anomaly detection can occur. The proposed approach is novel in its application to very large scale systems, and in particular, its use of contextual information to reduce the rate of false positives. Further, we posit that our work can be extended by defining a third step based on the semantic locality of the data, providing a further reduction in the number of anomalies which are false positive.

The following sections of the paper are organized as follows: Section II will describe related works in the field of anomaly detection in streaming sensor systems. Section III will outline the approach taken by the proposed research. The framework will be applied in a case study in Section IV. Finally, Section V will describe concluding thoughts and ideas for future work in this area.

II. Related Work

Anomaly detection algorithms can be broadly categorized as point anomaly detection algorithms and context-aware anomaly detection algorithms [1]. Contextual anomalies are found in datasets which include both contextual attributes and behavioral attributes. For example, environmental sensors generally include the sensor reading itself, as well as spatial and temporal information for the sensor reading. Many previous anomaly detection algorithms in the sensoring domain focus on using the sequential information of the reading to predict a possible value and then comparing this value to the actual reading. Hill and Minsker [6] propose a data-driven modelling approach to identify point anomalies in such a way. In their work they propose several *one-step ahead* predictors; i.e. based on a sliding window of previous data, predict the new output and compare it to the actual output. Hill and Minsker [6] note that their work does not easily integrate several sensor streams to help detect anomalies. This is in contrast to the work outlined in this paper where the proposed technique includes a contextual detection step that includes historical information for several streams of data, and their context. In an earlier work, Hill et al. [7] proposed an approach to use several streams of data by employing a real-time Bayesian anomaly detector. The Bayesian detector algorithm can be used for single sensor streams, or multiple sensor streams. However, their approach relies strictly on the sequential sensor data without including context. Focusing an algorithm purely on detection point anomalies in the sensoring domain has some drawbacks. First, it is likely to miss important relationships between similar sensors within the network as point anomaly detectors work on the global view of the data. Second, it is likely to generate a false positive anomaly when context such as the time of day, time of year, or type of location is missing. For example, hydro sensor readings in the winter may fluctuate outside the acceptable anomaly identification range, but this could be due to varying external temperatures influencing how a building manages their heating and ventilation.

Little work has been performed in providing context-aware anomaly detection algorithms. Srivastava and Srivastava [8], proposed an approach to bias anomaly detectors using functional and contextual constraints. Their work provides *meaningful* anomalies in the same way as a post-processing algorithm would, however, their approach requires an expensive dimensionality reduction step to flatten the semantically relevant data with the content data. Mahapatra et al. [9] propose a contextual anomaly detection framework for use in text data. Their work focuses on exploiting the semantic nature and relationships of words, with case studies specifically addressing *tags* and *topic* keywords. They had some promising results, including a reduction in the number of false positives identified without using contextual information. Their approach was able to use well-defined semantic similarity algorithms specifically for identifying relationships between words. This is in contrast to the work proposed in this paper as we are concerned with contextual information such as spatio-temporal relationships between sensors. Similar to the work proposed in this paper is their use of contextual detection as a post-processing step. This allows the algorithm to be compared and optimized at two distinct steps: point anomaly detection, and contextual anomaly detection.

Miller et al. [10] discuss anomaly detection in the domain of attributed graphs. Their work allows for contextual data to be included within a graph structure. One interesting result is that considering additional metadata forced the algorithm to explore parts of the graph that were previously less emphasized. A drawback of Miller et al.'s [10] work is that their full algorithm is difficult for use in real-time analytics. To compensate, they provide an estimation of their algorithm for use in real-time analytics, however the estimation is not explored in detail and so it is difficult to determine its usefulness in the real-time detection domain.

Other work has been done in computationally more expensive algorithms, such as support vector machines (SVMs) and neural networks. In general, these algorithms require a large amount of training time, and little testing time. In most cases this is acceptable as models can be trained in an offline manner, and then evaluated in real-time. One disadvantage to using these classification-based algorithms is that many require accurate labels for normal classes within the training data [5]. This is difficult in scenarios such as environmental sensor networks where there is little to no labelling for each sensor value. Shilton et al. [11] propose a SVM approach to multiclass classification and anomaly detection in wireless sensor networks. Their work requires data to have known

classes to be classified into, and then those data points which cannot be classified are considered anomalous. One issue that the authors present is the difficulty in setting one of the algorithm's parameters. In particular, changing the value has a direct impact on the rate in which the algorithm produces false negatives, or in which the algorithm detects true positives. To reduce the effect of the computational complexity of these algorithms, Lee et al. [12] have proposed work to detect anomalies by leveraging Hadoop. Hadoop is an open-source software framework that supports applications to run on distributed machines. Their work is preliminary in nature and mostly addresses concerns and discussion related to anomaly detection in Big Data. Another online anomaly detection algorithm has been proposed by Xie et al [13]. Their work uses a histogram-based approach to detect anomalies within hierarchical wireless sensor networks. A drawback to their approach is their lack of consideration for multivariate data. That is, their work focuses strictly on developing histograms for the data content but not the context of the data. The authors address this as future work, indicating that inclusion of contextual data would improve the generality and detection performance of their algorithm.

III. Contextual Anomaly Detection

The proposed technique is composed of two distinct components: the content anomaly detector and the contextual anomaly detector. The content anomaly detector will be discussed in Section III-A, and the contextual anomaly detector will be discussed in Section III-B. The primary reason for creating a separation of concerns between content and context is in the interest of scalability for large amounts of data. The content-based detector will be capable of processing every new piece of data being sent to a central repository as it will use an algorithm with a fast testing time. In contrary to this, the context-based detector will be used in two situations: to help determine if the anomaly detected by the content detector is a false positive, and to randomly ensure that the sensor is not producing wholly anomalous results. The latter reason being that a sensor may be acting non-anomalous within its own history of values, but not when viewed with sensors with similar context. Section III-B will also outline the MapReduce technique to train the *sensor profiles*, thus determining which sensors are contextually similar. We define here a *sensor profile* as a contextually aware representation of the sensor as a subset of its attributes. Broadly, comparing an incoming sensor value with the corresponding sensor profile consists of comparing the incoming value with an average of all the sensor values composing the corresponding sensor profile. A pictorial representation of the sensor profile concept is illustrated in Figure 1.

Algorithm 1 illustrates the process of the technique from a component-level. This algorithm corresponds with the diagram shown in Figure 2. The `UnivariateGaussianPredictor` function evaluates the sensor against historical values taken from the same sensor. The function will calculate a prediction based on the previous values and compare that prediction against the actual result. An anomaly will be flagged based on the distance between the actual value and the discovered value. This will be discussed in more detail in Section III-A. The `GetSensorProfile` function will request the other sensors that are contextually similar to the sensor being evaluated. `MultivariateGaussianPredictor` then compares the sensor value with a mean value from the sensors found in the sensor profile. Again, based on the result of this evaluation, the anomaly can be rejected as being anomalous or confirmed as being both a content and context-based anomaly. Another important note is the `IsRandomContextCheck` function which is part of the if-statement. This will determine whether a random, perhaps non-anomalous, sensor value be sent to the context-based detector. The reason for this is primarily to

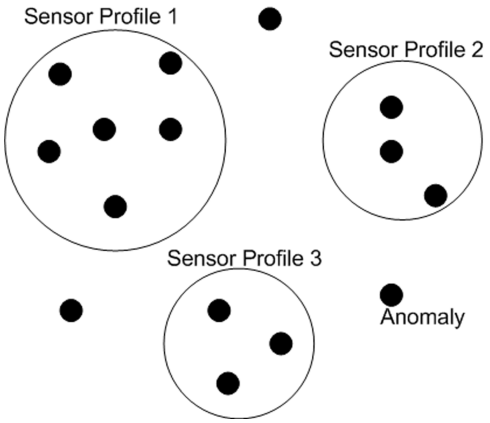


Fig. 1: Sensor Profile

input : $SensorValue$, $SlidingWindow$, $SensorHistory$
output: $Anomaly$

content ← `UnivariateGaussianPredictor`($SensorValue$, $SlidingWindow$, $SensorHistory$)

if `IsAnomalous` *(content)* || `IsRandomContextCheck` *()* **then**
  profile ← `GetSensorProfile` ($SensorValue$);
  context ← `MultivariateGaussianPredictor` ($SensorValue$, $profile$);
  if `IsAnomalous` *(context)* **then**
    return $Anomaly$=true;
  end
  else
    return $Anomaly$=false;
  end
end
else
  return $Anomaly$=false;
end

Algorithm 1: Contextual Anomaly Detection

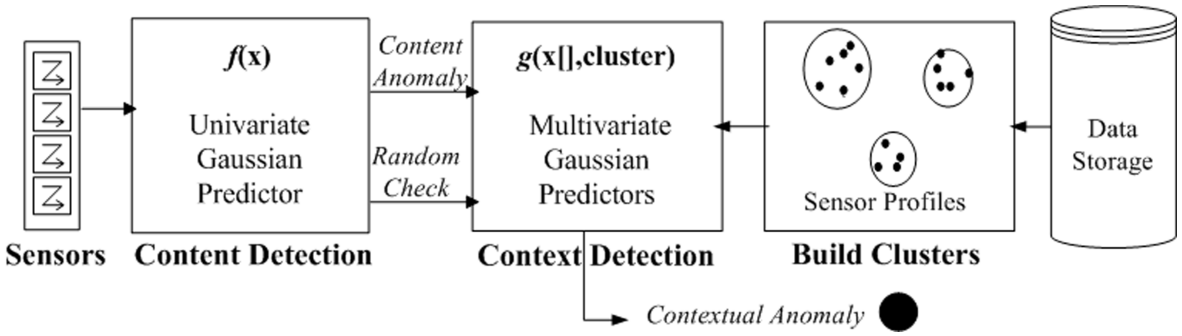


Fig. 2: Algorithm Overview

check whether the sensor is behaving anomalous with respect to the sensor profile.

A. Content Anomaly Detection

Content anomaly detection, or point anomaly detection, has been well explored in literature. In particular, the proposed content anomaly detection technique will use a *univariate Gaussian predictor* to determine point anomalies. Univariate Gaussian predictors build a historical model of the data, and then predict and compare new values based on the model. The predictor will be univariate in that it will only consider the historical sensor readings to adjust the parameters of the model. There will be no consideration for the contextual meta-information associated with the sensor readings. This ensures that the predictor can classify new values quickly while sacrificing some accuracy. Speed is the most important characteristic for the point detector as it needs to evaluate a high velocity and volume of data in real-time. The accuracy shortcoming will be handled by the contextual anomaly detector.

The univariate Gaussian predictor relies on defining two parameters during the training of the algorithm, μ and σ^2. Equation (1) and Equation (2) show how these two parameters are set, where m is the number of training values, and $x^{(i)}$ is the sensor reading for training value i. An anomaly is detected when $p(x) < \epsilon$, where ϵ is a threshold value set during implementation.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \tag{1}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2 \tag{2}$$

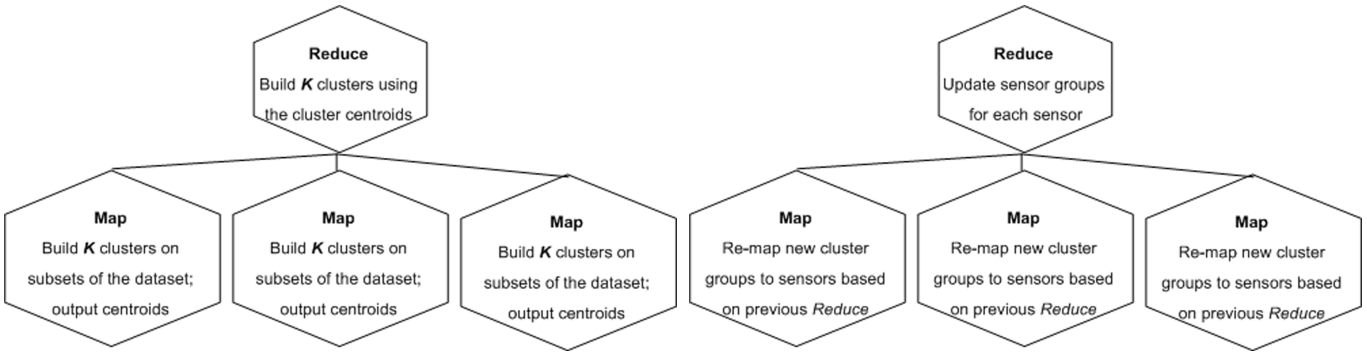
$$\begin{aligned} p(x) &= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{(x_j - \mu_j)^2}{2\sigma^2} \\ &= \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{(x - \mu)^2}{2\sigma^2} (\because n = 1) \end{aligned} \tag{3}$$

B. Contextual Anomaly Detection

The contextual anomaly detector is based on two concepts: defining the *sensor profiles* and assigning each sensor to one of the sensor profiles, and evaluating the current sensor value (declared anomalous by the content anomaly detector) against the sensor profile's average expected value. The sensor profiles are defined using a multivariate clustering algorithm; the algorithm is multivariate to include the sensors multidimensional contextual metadata which may include location, building, ownership company, time of year, time of day, and weather phenomena. The clustering algorithm will place each sensor within a sensor profile and then assign that profile group to the sensor. When a sensor has been declared anomalous by the content anomaly detector, the context anomaly detector will determine the average expected value of the sensor group. Then, in a similar way as in Equation 3, the context anomaly detector will determine whether the sensor value falls within the acceptable prediction interval.

The contextual anomaly detection algorithm will be learned offline, using a MapReduce [4] parallelism model to more efficiently perform on Big Data. MapReduce programs consist of two procedures *Map()* and *Reduce()*. The Map() procedure filters or sorts the data into manageable pieces, while the Reduce() procedure summarizes the results of the Map() procedures. In the contextual anomaly detection algorithm, the Map() procedure generates small clusters for partial sets of the sensor data. The Reduce() procedure takes the small clusters and aggregates them together to produce the final set of clusters. Concretely, both the Map() and Reduce() steps in Figure 3a use a clustering algorithm to determine cluster centroids. The Reduce() step, however, only uses the centroids determined by the Map() procedure, thus reducing the computational complexity exponentially.

The Map() and Reduce() steps in Figure 3b do not use a clustering algorithm. Instead, this MapReduce step is essentially re-mapping the cluster groups from the output of the first MapReduce step. For example, each initial Map() step will create *k*number of Map() calls* labels. The initial Reduce() step determines a set of *k* labels. It is the job of the second MapReduce procedure to map those *k*number of Map() calls* to the *k* labels. The first MapReduce process will use the k-means clustering algorithm [14]. K-means clustering aims to partition the dataset into a set of clusters that minimize the sum



(a) Build Initial and Overall Cluster Groups

(b) Re-Map Overall Groups to Initial Groups

Fig. 3: MapReduce for Determining the Cluster Groups

of squares distance between each data point, as per Equation 4. The k-means clustering algorithm will iterate through the following steps:

1) Randomly initiate K random clusters
2) Partition the dataset into the K random clusters, based on Equation 4; placing items into each cluster based on the smallest distance to the cluster
3) Re-calculate the centroids for each cluster
4) Repeat Step 2 until Step 3 does not modify cluster centroids

$$\min_{s} \sum_{i=1}^{k} \sum_{x_j} ||x_j - \mu_i||^2 \tag{4}$$

Once the clusters have been formed using k-means clustering, we define a Gaussian predictor for the *subset* of sensors which belong to each sensor profile. Then, each sensor profile has a specific Gaussian predictor which can be used to determine if a new sensor value is anomalous for that particular sensor profile family. Equations 5, 6, and 7 define the mean, sigma, and prediction function for the Gaussian predictor, where $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{nxn}$. Σ refers to the covariance matrix of the features used to define the Gaussian predictor. Also, $|\Sigma|$ refers to calculating the determinant of the covariance matrix. The new sensor values that are passed from the content anomaly detector are evaluated with respect to Equation 7. The value, *p(x)* is compared against some threshold, ϵ, and is flagged as anomalous if it is less than ϵ.

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} \tag{5}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T \tag{6}$$

$$p(x) = \frac{1}{(2\pi)^{(\frac{n}{2})} |\Sigma|^{(\frac{1}{2})}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \tag{7}$$

To summarize, the context anomaly detection algorithm proceeds as follows:

1) Offline: generate k clusters for each sensor profile using MapReduce
2) Offline: generate k Gaussian classifiers for each sensor profile
3) Online: evaluate corresponding Gaussian classifier when receiving a value by the content detector

C. Complexity and Parameter Discussion

One important aspect of any anomaly detection algorithm is its algorithmic complexity. The reason for selecting a clustering-based technique for the contextual anomaly detector is that it has a relatively short testing time, and a longer training time. For the contextual anomaly detector this is acceptable because training will happen offline and testing will happen relatively infrequently. More important is the content anomaly detector as the content detector test will run on *every* new sensor data. A major reason for selecting a univariate Gaussian predictor is its ability to evaluate new values quickly. This is because it is computationally cheaper than the multivariate counterpart.

To cope with higher computational complexity, the proposed algorithm also exposes parallelization and data reduction in two ways. First, the MapReduce portion uses a small set of the data to determine initial clusters. These clusters are then combined based on just the centroid information to produce a new set of clusters, and their corresponding centroids. This ensures that while the clustering algorithm may still be computationally expensive, the data is being reduced to $\log n$ number of Maps, thus reducing the number of tuples evaluated by each map to $\log n$. Similarly, the anomaly detection evaluation uses the notion of data reduction by only evaluating the computationally more expensive contextual anomaly detector on a very small subset of the data. This is based on the assumption that an anomaly is a rare occurrence. It is difficult to concretely define this reduction; however, if one can assume that the percentage of anomalous readings in a dataset is 0.001%, then the contextual detector need only consider that fraction of the initial data. The modularity of the content detector and the context detector allows the proposed research to be implemented in a large distributed environment. The content detectors can independently process information

in parallel on distributed machines and only need to share findings to the context detector when an anomaly is detected.

Another important aspect of the proposed algorithms is the selection of some of the parameters. For example, selection of the k in the k-means clustering algorithm will have a large impact on the accuracy of the Gaussian predictors. The problem of parameter selection is well-known in data mining but a work entitled *k-means++* by Arthur and Vassilvitskii [15] attempts to overcome this issue, specifically for k-means clustering. Therefore, the implementation of this work will utilize the k-means++ algorithm in place of the classic k-means algorithm. This does not change the sets of equations listed earlier in this section. Another set of parameters that are subject to discussion are the weights of the content anomaly detection algorithm. In the implementation we use a similar approach to k-means++ where the initial seeds are randomly distributed based on the average values of the sensor data. Additionally, the first few iterations also randomly reseed some values.

IV. Experimental Results

The preliminary evaluation of this work was done in conjunction with Powersmiths, a company specializing in providing sensor equipment to businesses to help build a sustainable future. In particular, Powersmiths provides products and services that reduce electricity waste, identify resource use on a building's sub-system level, manage greenhouse gas emissions, and coordinate corporate greening initiatives. To do this, Powersmiths uses sensor data that is pulled from the electrical, water, and gas systems within the business. This data is pushed to a cloud-based data storage solution where some analysis is completed and pushed back to the consumer. The evaluation of the proposed technique was done using their existing system. Preliminary studies were not done in real-time but rather trained in batch over their historical data and validated using a test dataset. To test the implementation offline, while emulating a real-time environment, several pseudo data streams were created and pushed to the detector at regular intervals mirroring the real world case. The following subsections will detail the implementation of the technique, and the results.

A. Powersmiths and the Dataset

Powersmiths is one of the leaders in sustainable power distribution products in Canada [16]. They provide energy metering and monitoring services through their proprietary physical devices and consumer-facing application *Windows on the World*. Powersmiths technologies can be found in a variety of business sectors; including: schools, hospitals, data centers, and building owners. The Powersmiths vision is two-fold: reducing electricity waste to improve the environment for future generations, and generating electricity savings for the customers. Currently, customers may have tens to hundreds of sensors, each with the ability to produce sensor data on the order of a few seconds, to several minutes. To cope with such a high volume and velocity of data, the company is currently expanding their Windows on the World application by porting many of their services *to the cloud*. The goal is to provide more computational resources to store and analyse their large amounts of data. Additionally, Powersmiths provides analytical tools, including a variety of data visualizations, to inform their consumers and help achieve the Powersmiths vision. Powersmiths would like to extend their existing analytical tools as they continue to service larger amounts of data; one such tool would be contextual anomaly detection.

Powersmiths collects a wide-range of data; including: sensor streams (as byte arrays) with temporal information, sensor spatial information, consumer profile information (i.e. name, location, etc), and multi-media including videos, images, and audio. For the purposes of anomaly detection, Powersmiths has provided a subset of their data which includes electricity sensor streams, and their related temporal and spatial context. A table of the data is shown in Table I, for each time input, there are corresponding values for each of the four sensors (labelled Sensor 1...Sensor 4) and the physical location of the sensor (labelled Sensor 1 Location...Sensor 4 Location). For the purposes of expanding the contextual information for the sensors, the *Time* feature has been discretized into two new features: *Day of the Week* and *Time of Day*. Time of Day is discretized into three values: 0 representing readings occurring outside normal, 9-5, office hours; 1 representing values occurring during morning, 9-1, office hours; and 2 representing values occurring during afternoon, 2-5, office hours. These features are shown below the break in Table I. One can consider the values for *Day of the Week*, *Time of Day*, and the set of locations as contextual information for the sensors; while the content itself is simply the sensor reading at the given time. The given Powersmiths dataset includes 101,384 tuples; where 85% were used for training, and 15% were used for testing. As mentioned earlier, the testing dataset has been modified to simulate a real-world streaming environment from multiple sensors. This was done by using the time stamp and individual sensors to produce four test datasets (one for each sensor). The simulation environment processes concurrent sensors streaming data in every 10ms. The authors acknowledge that a static write frequency is the optimal case and does not wholly represent a real-world simulation; however, the results can still show how the proposed research responds in such an environment.

Feature	**Domain**
Time	DD/MM/YYYY HH:MM
Sensor 1	0.00 - 100.00
Sensor 1 Location	[a-zA-Z0-9]
Sensor 2	0.00 - 100.00
Sensor 2 Location	[a-zA-Z0-9]
Sensor 3	0.00 - 100.00
Sensor 3 Location	[a-zA-Z0-9]
Sensor 4	0.00 - 100.00
Sensor 4 Location	[a-zA-Z0-9]
Day of the Week	0 - 7
Time of Day	0,1,2

TABLE I: Dataset and Feature Domains

B. Implementation Details and Results

The preliminary implementation for this work was completed in Java using the Weka [17] open-source data mining library for building the clusters. The implementation was performed over an old view of the Powersmiths dataset. The dataset included sensor readings from the past three years in one building. There were four sensors included in this dataset, all measuring the electricity from power meters located in different areas within the building. For example, there were sensors for the two research and development areas, existing on two floors. Also, there were sensors for the two administrative sectors of the building. Information on the location, as well as the sensor reading time as described in Table I were included. The implementation was trained over 85% of the 101,384 tuples within the dataset. Consequently, the final 15% was used to build a simulation environment to show the algorithms efficiency when dealing with a high velocity of data.

The initial Gaussian anomaly predictor using only content information (i.e. the data stream itself) was built using all the sensor values in the training dataset. This predictor is labelled as `Univariate Gaussian Predictor` in Algorithm 1 and Figure 2. For the purposes of this proof of concept, the parameters μ and σ^2, from Equation (1) and Equation (2), were determined iteratively. In further studies the authors would like to show relative speed-ups and trade-offs in parallelizing this step of the algorithm. Evaluating the sensor data was done in real-time as the simulator streamed in values. Given the low computational expense of this step, parallelization may not provide a large performance increase.

To build the contextual anomaly detector, the initial clusters and contextual anomaly detection parameters were also determined using a batch training model. We empirically determined that three clusters were appropriate for building a clustering model; beyond three the clustering algorithm saw little improvement. The cluster process is shown as `Build Clusters` in Figure 2. After determining the sensor profiles by clustering, a contextual Gaussian predictor was built for each contextual cluster, or *sensor profile* as defined in Section III. Algorithm 1 labels this as `Multivariate Gaussian Predictor`. Again, for the purposes of this proof of concept, the arrays of μ and Σ for each sensor profile were determined iteratively. This would need to be compared to a parallel elision as future work. For the purposes of preliminary evaluation the authors felt it was more important to show the benefits of the contextual detection contribution over the performance improvements in parallelizing the parameter selection.

The results of our work can be described in two steps. The first step is in the algorithms ability to determine the point anomalies in real-time. The second step is to determine which of these anomalies are contextually anomalous, as well as point anomalous. In Figure 4 the results of the point anomaly detector are shown. The upper portion of the figure, shaded in grey, shows those values which were determined to be

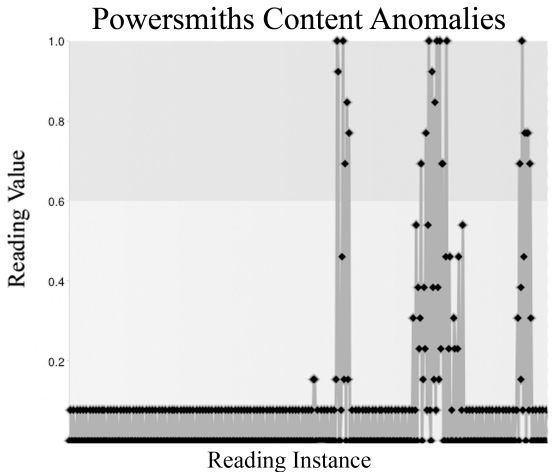


Fig. 4: Content Anomaly Detection

anomalous. Over the course of the simulation, 23 of the sensor values were considered to be point anomalies. Concretely, this means that 23 of the values were seen to be anomalous with respect to the expected value of that particular sensor, using only the sensor value. This means that less than 0.01% of the values were found to be anomalous, which is reasonable for the context of sensor streaming in a commercial building.

The second part of the results is determining which of those 23 sensor values are contextually anomalous. That is, based on the contextual information of: sensor physical locations, sensor reading time of day, sensor reading day of the week, and correlations between other sensors; which values remain anomalous? In Figure 5, a bar graph indicating the number of point anomalies, 23, and the number of contextually insignificant sensor values, 3. This figure outlines the reduction in the total number of anomalies detected by removing those that were considered contextually normal. The figure also illustrates that 13% of the potential point anomalies were cleared by the context detection process. Concretely, the approach determined that three of the point anomalies should not be considered as anomalous when including contextual information such as *Time of Day*, *Day of the Week*, and sensor spatial information.

Thus, we can say that the algorithm reduced the number of false-positive anomalies by 3. The other benefit we see here is that the more computationally expensive contextual detector only needed to evaluate 23 sensor readings, instead of the tens of thousands that are streamed to the point detector. Therefore, as the detection algorithm scales to more volumes of data, with higher velocities of data streams, the algorithm will still be able to evaluate the computationally more expensive contextual detector while still providing real-time detection.

V. Conclusions and Future Work

The work in this paper presents a novel approach to anomaly detection in *Big Sensor Data*. In particular, a contextual anomaly detection algorithm is proposed to enhance a point anomaly detection algorithm. To cope with the velocity and volume of Big Data, the anomaly detection algorithm relies on a fast, albeit less accurate, point anomaly detection algorithm

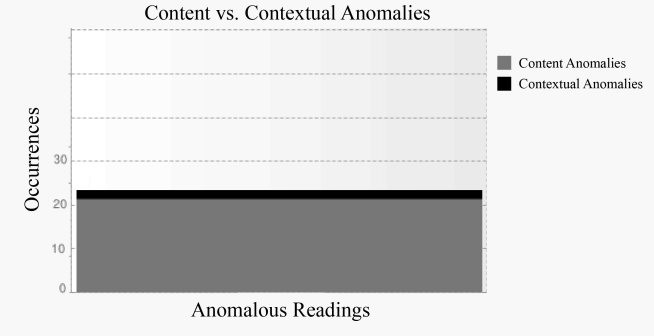


Fig. 5: Content vs. Contextual Anomalies

to find anomalies in real-time from sensor streams. These anomalies can then be processed by a contextually aware, more computationally expensive, anomaly detection algorithm to determine whether the anomaly was contextually anomalous. This approach allows the algorithm to scale to Big Data requirements as the computationally more expensive algorithm is only needed on a very small set of the data, i.e. the already determined anomalies.

The authors also propose a MapReduce approach to define the sensor profiles used in the context detector. The MapReduce approach uses an iterative k-means clustering algorithm to first determine k-clusters on a small subset of the data, and then combine these *k*number of Maps* clusters into just k-clusters, based on just the centroids from the initial Map() operations. This approach uses a similar thought paradigm as the anomaly detection algorithm: try to reduce the data for computationally expensive operations.

The preliminary results of this work are promising. The authors determined that the algorithm performs well in real-time, based on a simulation environment using real world data provided by Powersmiths, located in Brampton, Canada. The algorithm is able to find point anomalies in real-time, while also determining contextual anomalies, reducing the number of false positives in the point anomaly detector.

There are several areas for future work that are quite interesting. First, the dataset considered is only one type of Big Data; that is, *tall* datasets. It is important to consider other datasets, i.e. *wide*, which has a large number of features to consider. The authors believe this would be even more beneficial to this work as the point anomaly detector would still only consider a single value, whereas the contextual anomaly detector would receive an increase in ”context”. Second, the preliminary implementation of this work did not consider the random contextual anomaly check. One area of concern is that the point anomaly detector may occasionally mis-label a value as non-anomalous, where in fact it was contextually anomalous. Finally, the authors would like to explore further modules added to this work; for example, a semantic anomaly detector. Some applications include a domain ontology, or a description of the relationships between attributes. Leveraging these existing rules could further enhance the anomaly detection abilities of the proposed algorithm.

ACKNOWLEDGMENT

This research was supported in part by an NSERC CGS-M research grant to Michael Hayes at Western University (CGSM-444130-2013). Additionally, the authors would like to acknowledge the support provided by Powersmiths.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[2] Z. He, J. Z. Huang, X. Xu, and S. Deng, “Mining class outliers: Concepts, algorithms and applications,” in *Advances in Web-Age Information Management*, ser. Lecture Notes in Computer Science, Q. Li, G. Wang, and L. Feng, Eds. Springer Berlin Heidelberg, 2004, vol. 3129, pp. 589–599. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-27772-9_59

[3] Y. Kou and C. tien Lu, “Spatial weighted outlier detection,” in *In Proceedings of SIAM Conference on Data Mining*, 2006.

[4] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[5] S. Rajasegarar, C. Leckie, and M. Palaniswami, “Anomaly detection in wireless sensor networks,” *Wireless Communications, IEEE*, vol. 15, no. 4, pp. 34–40, 2008.

[6] D. J. Hill and B. S. Minsker, “Anomaly detection in streaming environmental sensor data: A data-driven modeling approach,” *Environ. Model. Softw.*, vol. 25, no. 9, pp. 1014–1022, Sep. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.envsoft.2009.08.010

[7] D. J. Hill, B. S. Minsker, and E. Amir, “Real-time bayesian anomaly detection in streaming environmental data,” *Water Resources Research*, vol. 45, no. 4, 2009. [Online]. Available: http://dx.doi.org/10.1029/2008WR006956

[8] N. Srivastava and J. Srivastava, “A hybrid-logic approach towards fault detection in complex cyber-physical systems,” in *Prognostics and Health Management Society, 2010 Annual Conference of the*, 2010, pp. 13–24.

[9] A. Mahapatra, N. Srivastava, and J. Srivastava, “Contextual anomaly detection in text data,” *Algorithms*, vol. 5, no. 4, pp. 469–489, 2012. [Online]. Available: http://www.mdpi.com/1999-4893/5/4/469

[10] B. Miller, N. Arcolano, and N. Bliss, “Efficient anomaly detection in dynamic, attributed graphs: Emerging phenomena and big data,” in *Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on*, 2013, pp. 179–184.

[11] A. Shilton, S. Rajasegarar, and M. Palaniswami, “Combined multiclass classification and anomaly detection for large-scale wireless sensor networks,” in *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on*, 2013, pp. 491–496.

[12] J. R. Lee, S.-K. Ye, and H.-D. J. Jeong, “Detecting anomaly teletraffic using stochastic self-similarity based on Hadoop,” in *Network-Based Information Systems (NBiS), 2013 16th International Conference on*, 2013, pp. 282–287.

[13] M. Xie, J. Hu, and B. Tian, “Histogram-based online anomaly detection in hierarchical wireless sensor networks,” in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, 2012, pp. 751–759.

[14] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Applied statistics*, pp. 100–108, 1979.

[15] D. Arthur and S. Vassilvitskii, “K-means++: The advantages of careful seeding,” in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA ’07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[16] Powersmiths, “Powersmiths: Power for the Future,” 2010, http://ww2.powersmiths.com/index.php?q=content/powersmiths/about-us.

[17] Machine Learning Group, “Weka 3 - Data Mining with Open Source Machine Learning Software in Java,” 2012, http://www.cs.waikato.ac.nz/ml/weka/.