Hayes and Capretz *Journal of Big Data* (2015) 2:2
DOI 10.1186/s40537-014-0011-y

Journal of Big Data
a SpringerOpen Journal

RESEARCH Open Access

Contextual anomaly detection framework for big sensor data

Michael A Hayes and Miriam AM Capretz*

*Correspondence:
mcapretz@uwo.ca
Department of Electrical and Computer Engineering, Western University, London, Canada

Abstract

The ability to detect and process anomalies for Big Data in real-time is a difficult task. The volume and velocity of the data within many systems makes it difficult for typical algorithms to scale and retain their real-time characteristics. The pervasiveness of data combined with the problem that many existing algorithms only consider the content of the data source; e.g. a sensor reading itself without concern for its context, leaves room for potential improvement. The proposed work defines a contextual anomaly detection framework. It is composed of two distinct steps: content detection and context detection. The content detector is used to determine anomalies in real-time, while possibly, and likely, identifying false positives. The context detector is used to prune the output of the content detector, identifying those anomalies which are considered both content and contextually anomalous. The context detector utilizes the concept of profiles, which are groups of similarly grouped data points generated by a multivariate clustering algorithm. The research has been evaluated against two real-world sensor datasets provided by a local company in Brampton, Canada. Additionally, the framework has been evaluated against the open-source Dodgers dataset, available at the UCI machine learning repository, and against the R statistical toolbox.

Keywords: Big data analytics; Contextual anomaly detection; Predictive modelling; Multivariate clustering; Streaming sensors

Introduction

Anomalies are abnormal events or patterns that do not conform to expected events or patterns [1]. Identifying anomalies is important in a broad set of disciplines; including, medical diagnosis, insurance and identity fraud, network intrusion, and programming defects. Anomalies are generally categorized into three types: point, or content anomalies; context anomalies, and collective anomalies. Point anomalies occur for data points that are considered abnormal when viewed against the whole dataset. Context anomalies are data points that are considered abnormal when viewed against meta-information associated with the data points. Finally, collective anomalies are data points which are considered anomalies when viewed with other data points, against the rest of the dataset.

Algorithms to detect anomalies generally fall into three types: 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*, and to those that rely only partially on external input. 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 Data*, and in particular, 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. For example, in Web logs, anomaly detection can be used to identify abnormal behavior, such as identify fraud. In many of these areas one difficulty is coping with the velocity and volume of the data while still providing real-time support for detection of anomalies. Further, future prediction, energy usage reduction strategies, and anomaly detection are popular sources of new technological developments, many aimed at creating intelligent buildings. Intelligent builds are those that can manage, optimize, and reduce their own energy consumption, based on Big sensor Data [2].

There is also much discussion on the types of algorithms applied to anomaly detection; some consider that there is a paradigm shift in the types of algorithms used: from computationally expensive algorithms to computationally inexpensive algorithms [3]. The inexpensive algorithms may have much higher training accuracy error; that is, the error accumulated per record when training. However, when normalized over the entire, large, dataset, the higher training accuracy error converges to a lesser prediction error. Prediction error is defined as the error accumulated when predicting new values from a trained predictor [4]. The prediction error for the inexpensive algorithm is within similar ranges as those found with the computationally more expensive algorithm, yet occurring over a much smaller time frame. A motivation of this work is to then take this notion and shift it to incorporate these computationally expensive algorithms which still generally perform better.

Contextual anomaly detection seeks to find relationships within datasets where variations in external behavioural attributes well describe anomalous results in the data. For example, viewing data in the context of time, or in the context of time-related concepts such as seasons, weekdays and weekends, workdays and time-off, can reveal anomalous behaviour directly correlated with such context. This is distinct from content anomalies which can be defined as abnormal instances in data with respect to the implicit data alone. An example of a content anomaly is an abnormal spike in user logins on a website, independent from external reasons. An example for contextual anomalies can be found in a use-case such as power consumption likely has context-based, time-related relationships: it makes sense to posit that the power consumption of an office building is much higher during midday, during a work day, compared to at night, during a weekend. One aspect of this work is to explore, and show the work is successfully applied, for these more obvious relationships while also remaining expandable to learning and revealing complex contextually anomalous behaviours.

Some related works have focused on anomaly detection in data with spatial relationships [5], while others propose methods to define outliers based on relational class attributes [6]. 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. Additionally, 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 [7]. 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 [8].

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. 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: the "Background and literature review" section will describe related works in the field of anomaly detection in streaming sensor systems. The "Research design and methodology" section will outline the approach taken by the proposed research. The framework will be applied on three datasets in the "Results and discussion" section. Finally, the "Conclusions" section will describe concluding thoughts and ideas for future work in this area.

Background and literature review

Anomaly detection is involved in a variety of applications, across many disciplines. As such, terminology and background will be introduced to facilitate understanding for the rest of the paper. Anomaly detection algorithms can be categorized as point detection, collective detection, or context-aware detection algorithms [1]. Contextual detection is the root of the work presented in this paper and so will be the focus on much of the related works. Contextual anomalies exist where the dataset includes a combination of *behavioural* and *contextual* attributes. These terms are also defined as *environmental* and *indicator* attributes, as introduced by Song et al. [9]. Behavioural attributes are attributes such as the sensor reading itself. Contextual attributes take on one of four forms, defined in Table 1.

Table 1 Definitions for types of contextual attributes

Term	Definition
Spatial	The records in the dataset include features which identify locational information for the record. For example, a sensor reading may have spatial attributes for the city, province, and country the sensor is located in; it could also include finer-grained information about the sensors location within a building, such as floor, room, and building number.
Graphs	The records are related to other records as per some graph structure. The graph structure then defines a spatial neighbourhood whereby these relationships can be considered as contextual indicators.
Sequential	The records can be considered as a sequence within one another. That is, there is meaning in defining a set of records that are positioned one after another. For example, this is extremely prevalent in time-series data whereby the records are timestamped and can thus be positioned relative to each other based on time readings.
Profile	The records can be clustered within profiles that may not have explicit temporal or spatial contextualities. This is common in anomaly detection systems where, for example, a company defines profiles for their users; should a new record violate the existing user profile, that record is declared anomalous.

Contextual anomaly applications are normally handled in one of two ways. First, the context anomaly problem is transformed into a point anomaly problem. That is, the application attempts to apply separate point anomaly detection techniques to the same dataset, within different contexts. In this approach, it is necessary to define the contexts of normal and anomalous records apriori, which is not always possible within many applications. This is true for the Big sensor Data use case, where it is difficult to define the entire set of known anomalous records. The second approach to handling contextual anomaly applications is to utilize the existing structure within the records to detect anomalies using all the data concurrently. This is especially useful when the context of the data cannot be broken into discrete categories, or when new records cannot easily be placed within one of the given contexts. The second approach generally requires a higher computational complexity than the first approach as the underlying algebra in calculating a contextual anomaly is computationally expensive.

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 [10] 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 [10] 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. [11] 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 [12], 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. [13] 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.

A different approach for contextual detection is that work of AlEroud et al. [14], who apply contextual anomaly detection to uncover zero-day cyber attacks. Their work involves two distinct steps, similar to the modules described in this paper: contextual misuse module, and an anomaly detection technique. There are other minor modules, such as data pre-processing, and profile sampling. The first major component, contextual misuse, utilizes a conditional entropy-based technique to identify those records that are relevant to specific, useful, contexts. The second component, anomaly detection, uses a 1-nearest neighbour approach to identify anomalies based on some distance measure. This component is evaluated over the records individually to determine whether connections between records indicate anomalous values. The work presented by AlEroud et al. [14] is similar to the work presented in this paper in that the detection is composed of two distinct modules. However, the content component of their work involves calculating difficult distance measures that are not always easily definable. For example, when faced with many features that each have different data types or domains, it is difficult to calculate suitable distance metrics as finding a common method to aggregate the features is also difficult. Another drawback is that each module is normally evaluated for all new incoming values. While the authors do say that the first component aims to reduce the dimensionality required for the second component, they go on to mention that both the contextual component and anomaly detection component are calculated individually to evaluate the anomaly detection prowess of the approach.

Miller et al. [15] 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 [15] 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 [8]. This is difficult in scenarios such as environmental sensor networks where there is little to no labelling for each sensor value. Shilton et al. [16] 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. To reduce the effect of the computational complexity of these algorithms, Lee et al. [17] 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. [18]. 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.

Another component of the work presented in this paper is deploying a modular, hierarchical, framework to ensure the algorithm can cope with the velocity and volume of Big Data. Other work by Kittler et al. [19] present a system architecture to detect anomalies in the machine perception domain. In particular, they propose a set of definitions and taxonomies to clearly define the roles and boundaries within their anomaly detection architecture. Their work is underlined with a Bayesian probabilistic predictor which is enhanced by concepts such as outlier, noise, distribution drift, novelty detection and rare events. These concepts are used to extend their application to other domains that consider similar concepts. This approach is in distinct contrast to the work proposed by this paper. Concretely, Kittler et al. [19] present context as more analagous to *semantics* in that the additional information they add is similar to domain ontologies rather than contextual information inherently associated within the data.

Research design and methodology

The work proposed in this paper describes a framework consisting of two distinct components: the content anomaly detector and the contextual anomaly detector. The work is described as a framework as it provides an extendible and modular approach to anomaly detection, not requiring specific implementations for each module: the content and context detectors in particular. The rest of this paper will propose one possible solution for the modules, which works particularly well for the streaming sensor use-case.

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 contrast 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. Sub-section "Contextual anomaly detection" will also outline the 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. 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. This creates two levels of abstraction for the contextual detector. First, since similar sensors are combined into sensor profiles, a level of context is created at a sensor attribute level. Second, the contextual detection uses a correlation matrix of the attributes for the sensors to further apply context to incoming values for comparison. The use of clustering is also used for two purposes. First, clustering helps reduce the population size by creating sub-tasks that run against the computationally expensive contextually anomaly detection function. Second, clustering provides early insight into the context of the new sensor value. Again, this early context picture is further enhanced when the contextual detector is applied. 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. 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. This algorithm corresponds with the diagram shown in Figure 2. The `GetSensorProfile` function will request the other sensors that are contextually similar to the sensor being

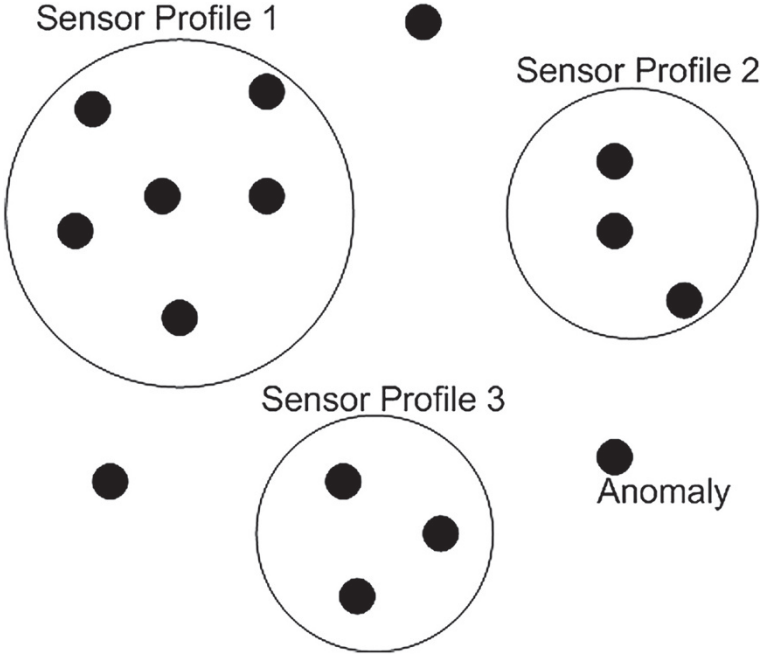

Figure 1 Sensor Profile. Definition of a sensor profile with respect to anomalies, and groups of similar sensors. Each point represents a sensor reading; circles indicate groups of sensors, that is, sensor profiles.

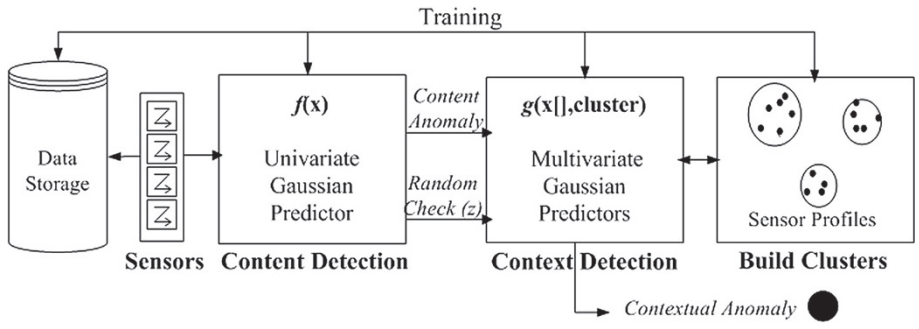

Figure 2 Algorithm Overview. High-level overview of the algorithm showing the three different modules and how they interact with one another. Additionally shows the input and output for each module.

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

Algorithm 1: Contextual Anomaly Detection

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

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.

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

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

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

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 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.

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

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

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

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

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

Complexity and parameter discussion

The contextual detection algorithm works on multiple dimensions of the dataset, and thus has a higher computational complexity in comparison to the content detector. To cope with higher computational complexity, the proposed framework utilizes parallelization and data reduction in an important way. 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. 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. Then, the framework can horizontally scale by adding additional machines to indepdently process new data.

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 [20] 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.

Results and discussion

The preliminary evaluation of this work was primarily done in conjunction with Powersmiths, a company specializing in providing sensor equipment to businesses to help build a sustainable future [21]. Figure 3 illustrates a general sensor streaming application. 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. 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.

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 sub-sections will detail the implementation of the technique, and the results. The evaluation of the proposed work will also be shown on the open-source Dodgers dataset, available from the UCI Machine Learning repository [22,23].

Powersmiths and the Datasets

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 temperature sensor streams, and their related temporal and spatial context. A table of the electrical data is shown in Table 2, for each time input, there are corresponding values for each of the four sensors (labelled Sensor

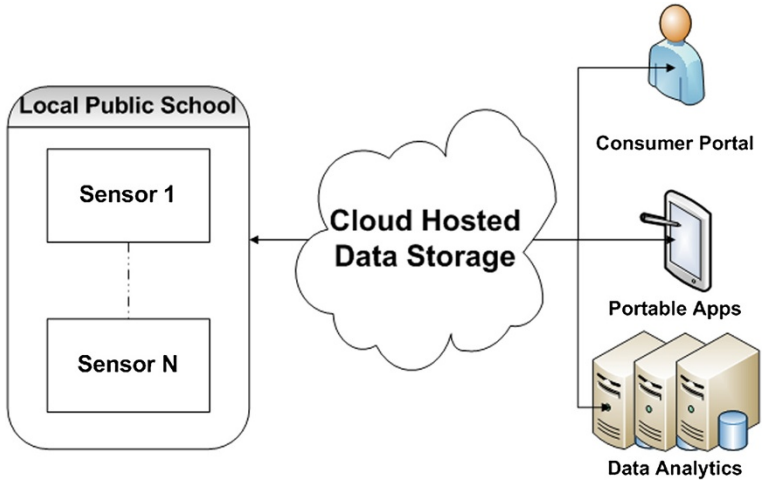

Figure 3 Big Sensor Data Case Study. Introduction to the case study used for evaluation purposes. The illustration diagrams out the relationship between the end-users and the raw sensor information.

Table 2 Dataset and feature domains

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

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 2. 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 for the electrical 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. The second dataset, for temperature streaming sensor readings, consists of 26,943 readings. Again, for testing and evaluation purposes this dataset was split into 85% for training, and 15% for evaluation. The dataset consists of five sensors; their meta-information is shown in Table 3. The dataset consists of five sensors that are recording temperature readings, in degrees Celsius. The sensors are again distributed across the building's main office; both recording values of rooms themselves, as well as latent temperatures of the walls. Like dataset 1, there is additional contextual information that can be extracted from the *Time* attribute, as shown in Table 4.

The implementation for the framework was completed using a virtualized sensor stream. That is, the application was built around an old view of the *Big Sensor Data* dataset, and not done over the real, live, data. To ensure the data was still tested as though it was real-time, a virtualized sensor stream was created. This was accomplished by extracting the different sensors from the entire dataset into their own individual datasets. The queue would then be incrementally evaluated and removed. A code listing for this process can be found in Listing 1. Another important implementation detail is the determination in the number of sensor profiles to create during the contextual detection

Table 3 Sensor dataset 2: temperature

Location	Sensor location	Associated	Unit
Main	Head Office	Forge Hallway	Celsius
Main	Head Office	Forge Room	Celsius
Main	Head Office	IT Closet	Celsius
Main	Head Office	South Wall	Celsius
Main	Head Office	North Wall	Celsius

process. Determining the number of profiles was done as a pre-processing step, iteratively increasing the number of sensor profiles until the accuracy was no longer improved. Empirically this was determined to be three clusters for the Powersmiths datasets. Logically this also made sense as Powersmiths provides three types of sensoring to their clients.

Listing 1 Code Snippet for Virtualized Sensor Stream

```
//Instantiate Queue
Queue virtualQueue = new LinkedList();
...
//Concurrently write new sensor values
synchronized(virtualQueue) {
   virtualQueue.add(sensorValue);
}
...
//Evaluate the top-most queue against
//the real-time model
while(true) {
  if(virtualQueue.poll() != null)  {
    String val[] = virtualQueue.remove();
    boolean isAnomalous =
      checkValueWithRealTime(
    Double.parseDouble(val[1]));
    if (isAnomalous) anomalyArray.add(
                value);
  }
}
```

Dataset 1 implementation and results: HVAC

The preliminary implementation for this work was completed in Java using the Weka [24] open-source data mining library for building the clusters. There were four sensors included in the HVAC dataset, all measuring the electricity from power meters located in different areas within the building. For example, there were sensors for the two research

Table 4 Dataset example

Date Time	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Day	Time of Day
03/01/2011 09:00	20.86	22.32	26.69	2.27	1.78	2	0
03/01/2011 10:00	20.32	21.63	26.05	0.88	-2.14	2	0
03/01/2011 11:00	20.11	21.62	25.87	-2.44	-3.24	2	0

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 2 were included.

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.

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 the "Research design and methodology" section. 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.

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

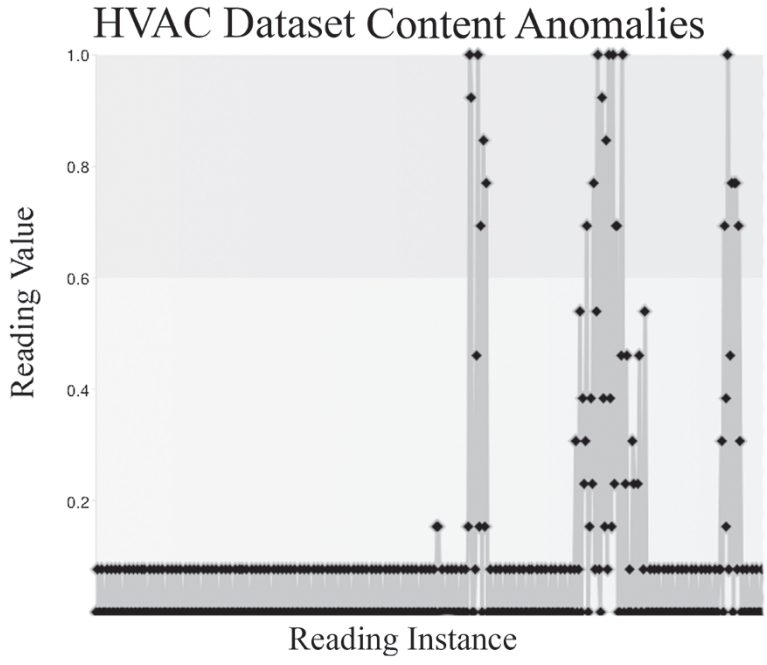

Figure 4 Content Anomaly Detection. The results of the proposed work on the HVAC sensor dataset. Each black circle represents a sensor reading, the readings are interconnected with dark grey lines. Additionally, the points highlighted in grey at the top of the image are those which have been identified as anomalous with respect to their content alone.

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 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 pie graph indicating the number of point anomalies, 23, and the number of contextually insignificant sensor values, 2. 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 9% of the potential point anomalies were cleared by the context detection process. Concretely, the approach determined that two 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 2. 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.

Dataset 2 implementation and results: temperature

Synonymous to dataset 1, the first component to evaluate is the content detection for the temperature dataset. In testing the proposed work's content detection using the test data, the content detector was able to find 254 anomalies. These anomalies are considered to be point anomalous. This is strictly larger than the previous dataset results; however, when taking a closer look at those values labelled as anomalous, the readings are consistent with

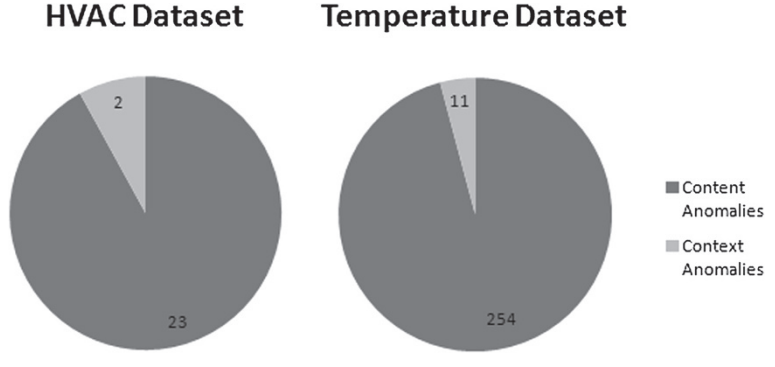

Figure 5 Context Detection for Datasets 1 and 2. Comparison between the two datasets for content and contextual anomalies identified.

a failed sensor. These results are shown in Figure 6. More discussion on this will occur after addressing the context detection component.

The second major component to be evaluated is the context detection. To perform context detection, the proposed framework first needs to determine the sensor profiles for the temperature sensors. For dataset 2, it was determined that two sensor profiles existed within the dataset; adding more clusters did not increase the effectiveness of the context detector. Further, the two sensor profiles that were revealed included one group for the two temperature sensors that record latent temperatures, and three sensors that record room temperature. Once determining the two sensor profile groups, two multivariate Gaussian predictors were trained, one for each cluster. The results of the contextual detector determined that 11 of the 254 point anomalies could be cleared as being non-anomalous with respect to their context. It was also found that when training the context detector with one sensor profile, i.e. not including the initial context of the data, there were no contextual anomalies cleared. Additionally, when removing the *Day* and *TimeOfDay*(TOD) attributes, no contextual anomalies were found. This further shows that the addition of the context detector has a positive impact of determining anomalous readings. The results of running the simulation using the contextual detection with the content detection for both datasets are shown in Figure 5. The figure shows the distribution of context anomalies found within the content anomalies.

The results for dataset 2 were also promising with anomalous examples shown in Table 5. The framework efficiently determined content based anomalies in real-time using the virtualized sensor streaming implementation. Details on the running times for the components is shown in Table 6.

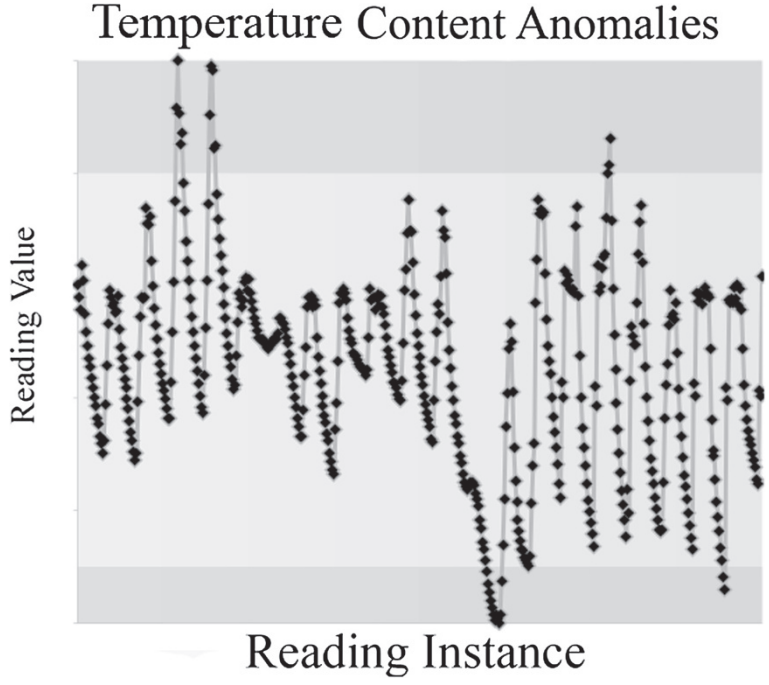

Figure 6 Content Detection for Dataset 2. The black diamonds represent reading instances. The top and bottom highlighted portions in grey identify reading instances that were considered anomalous with respect to their content for the temperature dataset.

Table 5 Anomalous examples for dataset 2

Sen. 1	Sen. 2	Sen. 3	Sen. 4	Sen. 5	Day	TOD
15.12	15.25	18.71	5.35	3.71	7	1
17.42	19.09	24.53	1.05	3.56	3	0
17.22	18.18	23.18	4.08	6.88	1	0
0	0	0	0	-1.26	4	2
0	0	0	0	-1.43	7	0
0	0	0	0	-1.47	7	0

Dataset 3 implementation and results: dodger loop

The final dataset used for evaluation is based on data collected by the Freeway Performance Measurement System (PeMS) in California. The data was collected for the *Dodgers Loop* in Los Angeles, for times when the Los Angeles Dodgers were playing baseball games. The initial goal of the data was to predict days when there were Dodgers games, based on the traffic seen at the Dodgers Loop. The data includes observations over 25 weeks, at 288 time slices per day. In total, there are three attributes, two contextual, one behavioural, and 50,400 tuples.

Synonymous to the Powersmiths datasets, the first component to evaluate is the content detection for the Dodgers Loop dataset. In testing the proposed work's content detection using the test data, the content detector was able to find 17 anomalies. These anomalies are considered to be point anomalous. These anomalies were *all* injected into the dataset as there were no anomalies found when initially running the algorithm. The authors attempted to inject values that should be considered anomalies (i.e. abnormally high amounts of traffic for early mornings where there was certainly no Dodger baseball game).

The second major component to be evaluated is the context detection. To perform context detection, the proposed framework first needs to determine the profiles for the Dodger Loop readings. For dataset 3, it was determined that four profiles should exist within the dataset; adding more clusters did not increase the effectiveness of the context detector. Once determining the four profile groups, four multivariate Gaussian predictors were trained, one for each cluster. The results of the contextual detector determined that 1 of the 17 point anomalies could be cleared as being non-anomalous with respect to their context. It was also found that when training the context detector with one sensor profile, i.e. not including the initial context of the data, there were no contextual anomalies cleared. Additionally, when removing the *Day* and *TimeOfDay* attributes, no contextual anomalies were found. This further shows that the addition of the context detector has a positive impact of determining anomalous readings, and is akin to earlier results for the Powersmiths data. Details on the running times for the components is shown in Table 7. From the table: the framework can successfully manage to detect anomalies in real-time, only using an average of 1432ns per content evaluation. Table 8 show the anomalies identified by the contextual predictor as well as the content predictor.

Table 6 Dataset 2 running time results

Component	Time
ReadCSV	1.145 sec
BuildContextModel	2.343 sec
BuildRealtimeModel	0.021 sec
CheckRealTime (average)	933 ns

Table 7 Dataset 3 running time results

Component	Time
ReadCSV	1.145 sec
BuildContextModel	0.063 sec
BuildRealtimeModel	0.011 sec
CheckRealTime (average)	1432 ns

Validation and cross-validation

We present the validation of our work in two ways. First, we discuss our work in comparison with other views on the same datasets we used, as well as other works in the area of anomaly detection. Second, we discuss our work with respect to Powersmiths use-case, and specifically related to the apriori knowledge they have provided. First, it is difficult to compare anomaly detection algorithms as the definition for an anomaly is highly dependent on the use-case. For example, Powersmiths may consider abnormal values within 10% deviation an anomaly, whereas another energy management company considers values within 8% deviation an anomaly. With this in consideration, we present a major benefit of our work: a sliding detection threshold. That is, the threshold for an anomaly to be considered abnormal can be configured dynamically based on the use case. A second way in which we validated our work is to compare it with the standard R statistical toolbox, and specifically the *outlier* function in the *outliers* package.

The previous evaluation sections attempted to provide insight into the validity of the proposed work with respect to real-world data. This section will further compare the results of the earlier sections to results from the popular **R** application for statistical computing. Figure 7 shows the results of running the outliers package on dataset 2. The largest section indicates the highest density statistical probability for dataset 2, while the other two dense sections indicate lesser populated areas. The anomalies are noted by black numbers; as seen in Figure 7, there are anomalous values in both sections. A summary of the comparison between the work presented in this paper and the **R** outliers package is shown in Table 9. Concretely, we can conclude that our proposed framework performed as well as the R outliers package in detecting the anomalies, and is much more scalable, and applicable to real-time detection.

When given thess datasets, Powersmiths mentioned that there were lengths of time when various sensors failed at their headquarters. Specifically in the temperature dataset, where the framework was able to successfully identify these anomalous readings, as shown below the double-line break in Table 5. This is promising as it validates the approach for the real-world data, knowing a set of values that were previously considered anomalous by Powersmiths. The detector additionally determined a set of anomalies that were not purely based on a totally failed sensor; these are shown above the double-line in Table 5. One of the difficulties with validating any anomaly detection approach is that the definition of an anomaly changes depending on the owner of the dataset, and their view over what should be considered anomalous. Therefore, a first step in validation is determining that the framework could initially identify all values which were considered anomalous by the dataset owner: Powersmiths themselves. Another consideration is performing cross-validation, that is, using different subsets of the dataset for training and

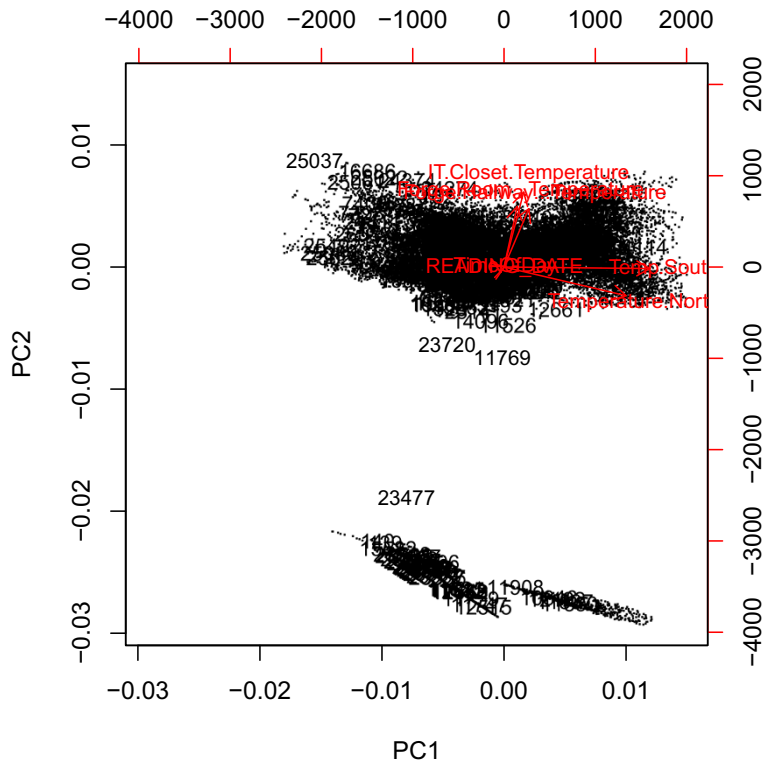

Figure 7 R Results. Graphical representation of the comparison with the R statistical toolbox. Specifically, the numbered points are those that were identified as anomalous with respect to the context of the sensor readings.

testing to ensure that the framework still identifies those values as anomalous in different subsets of the dataset.

Cross-validation was performed for the temperature dataset and found that the anomalous values presented by Powersmiths were again detected. In fact, one of the subsets for cross-validation included a subset of 15% of the dataset where there were *no* anomalies identified by Powersmiths. Indeed, the framework did not detect any anomalies for this subset of the data. To summarize the validation of the framework for the presented datasets, the approach was four-fold:

- Confirm that the framework is able to identify anomalies with respect to those identified by Powersmiths, the dataset owner, themselves.
- Validate that the framework is able to identify injected anomalies by the author; these are values that fall well out of the parameters of one or more of the attributes in the dataset.

Table 8 Anomalous Examples for dataset 3

Time	Day Of Week	Cars Per 5 min	Time Of Day
2760	3	76	0
3060	3	44	0
2820	4	0	0
0	0	0	0
540	0	-8	0

Table 9 Results comparison with the R outliers package

Concept	Discussion
Running time	A major difference between the work presented in this paper and the R outliers package is in the running time of both algorithms. One of the major design considerations for the proposed work was the ability to run in real-time. The R outliers package runs in batch over the entire set of data and runs on the order of minutes to completion, whereas our work runs on the order of seconds.
Anomalies found	The work presented in this paper and R outliers package detected a number of equivalent anomalies. The huge list of anomalies pertaining to the failed sensors identified apriori by Powersmiths were found by both detectors. Further, the work presented in this paper detected all of the anomalies detected by the R outliers package, including those that were contextually cleared.

- Compare the results with the R statistical toolbox to ensure that the results found by the framework are consistent with the offline approach of the R statistical toolbox.
- Cross-validate the results of the dataset with other subsets of the dataset. That is, use different subsets of 15% to compare and confirm that the framework still identifies all the anomalous values.

Random positive detection: implementation and results

In addition to implementing the framework's content and context detection, and evaluating three large datasets using each module, we have also explored the random positive detection module. The first approach we used was implementing a naive random detection module, which simply randomly passed *any* value that was flagged to be passed to the context detector. This approach was used with an implementation-defined random value, 0.01%. However, initial results for this approach proved the *base rate fallacy*: essentially outlining that our initial assumption for the random detection still being able to detect the anomalies is false. In other words, following the Bayes theorem, attempting to detect anomalies using this form of random detection has the following property:

$$P(anomaly|rand) = \frac{P(rand|anomaly) * P(anomaly)}{P(rand)} \tag{8}$$

From Equation 8, we can see that the probability that we randomly detect an event that already has a small percentage of occurring is very low. Due to this property, it is necessary to rethink the approach in determining the random values. Before continuing our approach for these three datasets, we attempted to determine, offline, whether there were any values to randomly detect. Concretely, we tested whether there were any contextual anomalies that were not passed to the context detector from the content detector. In running *only* the context detector over the entire test dataset, the results showed that there were no context anomalies that were not passed to the content detector. Therefore, for the three datasets used for the evaluation of this work, none contained anomalies that were normal with respect to their content, but abnormal with respect to their context.

Conclusions

The work presented in the paper describes a novel framework for anomaly detection in Big Data. Specifically, the framework utilizes a hierarchical approach to identify

anomalies in real-time, while also detecting a number of false positives. Then, a contextual anomaly detection algorithm is used to prune the anomalies detected by the content detector, but using the meta-information associated with the data points. 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 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 evaluation of the framework was also discussed based on the implementation details provided in this paper. The evaluation of the framework was performed using three sets of data; one for a set of HVAC electricity sensors, one for a set of temperature sensors, and a third set for a traffic system in California. The evaluation provided some conclusions of the work:

- The framework was able to positively detect, in real-time, content based anomalies for the datasets. Further, the context detector was able to determine some anomalies that should not be considered anomalous when evaluated with respect to context.
- The framework was able to positively detect anomalies that were determined apriori by the owner of the datasets. In particular, the framework determined a large set of anomalous readings which occurred when Powersmiths indicated a massive sensor failure.
- The framework performed competitively with the R outlier statistical package. The framework performed in real-time, in comparison with the batch approach provided by the R application.

One of the major goals of the framework was to remain modular and scalable for future works. As a result, there are several areas of future work that would be interesting to explore:

- The datasets considered in the evaluation section are only one type of Big Data: *tall* datasets. It is important to consider the other common type of application: *wide* datasets. These are attributed by a large number of features, with a smaller number of records. One of the major benefits of this work is that the hierarchy of content to contextual detection ensures that a computationally inexpensive algorithm reduces the number of records evaluated by the contextual detector.
- The framework has been tested in the simulated, virtualized, sensor streaming environment. Therefore, a logical future work would be to implement the framework within a working business environment that is streaming live data to the central repository.
- The framework could also be integrated with decision making systems within a live environment. The output of the framework is an indication that a sensor is acting anomalous. This information can be exploited by a decision making algorithm, such as a complex event processing framework, to coordinate changes within the business environment.

- The modularity of the framework allows further components to be added, or updated, in the framework. This introduces two such avenues of future work: first, the proposed modules can be modifed and updated with other types of algorithms. Second, additional modules could be added to the framework itself. For example, a semantic detection module can be included.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
MAH is the principal researcher for the work proposed in this article. His contributions include the underlying idea, background investigation, initial drafting of the article, and results implementation. MAMC guided the initial research idea, and played a pivotal role in editing the article. Both authors read and approved the final manuscript.

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

Received: 5 August 2014 Accepted: 10 December 2014
Published online: 27 February 2015

References
1. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):1–58
2. Chan H, Chou P, Duri S, Lei H, Reason J (2009) The design and implementation of a smart building control system. In: e-Business Engineering, 2009. ICEBE '09. IEEE International Conference On. IEEE. pp 255–262
3. Fan J, Han F, Liu H (2014) Challenges of big data analysis. Natl Sci Rev. Oxford University Press
4. Dalessandro B (2013) Bring the noise: Embracing randomness is the key to scaling up machine learning algorithms. Big Data 1(2):110–112
5. He Z, Xu X, Huang JZ, Deng S (2004) Mining class outliers: concepts, algorithms and applications in crm. Expert Syst Appl 27(4):681–697
6. Kou Y, Lu C-T (2006) Spatial weighted outlier detection. In: Proceedings of SIAM Conference on Data Mining. SIAM
7. Dean J, Ghemawat S (2008) MapReduce: Simplified data processing on large clusters. Commun ACM 51(1):107–113
8. Rajasegarar S, Leckie C, Palaniswami M (2008) Anomaly detection in wireless sensor networks. Wireless Commun IEEE 15(4):34–40
9. Song X, Wu M, Jermaine C, Ranka S (2007) Conditional anomaly detection. Knowl Data Eng IEEE Trans 19(5):631–645
10. Hill DJ, Minsker BS (2010) Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. Environ Model Softw 25(9):1014–1022
11. Hill DJ, Minsker BS, Amir E (2009) Real-time bayesian anomaly detection in streaming environmental data. Water Resources Res 45(4)
12. Srivastava N, Srivastava J (2010) A hybrid-logic approach towards fault detection in complex cyber-physical systems. In: Prognostics and Health Management Society, 2010 Annual Conference of The. IEEE. pp 13–24
13. Mahapatra A, Srivastava N, Srivastava J (2012) Contextual anomaly detection in text data. Algorithms 5(4):469–489
14. AlEroud A, Karabatis G (2012) A contextual anomaly detection approach to discover zero-day attacks. In: Cyber Security, 2012 International Conference On. IEEE. pp 40–45
15. Miller BA, Arcolano N, Bliss NT (2013) Efficient anomaly detection in dynamic, attributed graphs: Emerging phenomena and big data. In: Intelligence and Security Informatics (ISI), 2013 IEEE International Conference On. IEEE. pp 179–184
16. Shilton A, Rajasegarar S, Palaniswami M (2013) 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. IEEE. pp 491–496
17. Lee JR, Ye S-K, Jeong H-DJ (2013) Detecting anomaly teletraffic using stochastic self-similarity based on Hadoop. In: Network-Based Information Systems (NBiS), 2013 16th International Conference On. IEEE. pp 282–287
18. Xie M, Hu J, Tian B (2012) Histogram-based online anomaly detection in hierarchical wireless sensor networks. In: Trust, Security and Privacy in Computing and Communications, 2012 IEEE 11th International Conference On. IEEE. pp 751–759
19. Kittler J, Christmas W, Campos TD, Windridge D, Yan F, Illingworth J, Osman M (2013) Domain anomaly detection in machine perception: a system architecture and taxonomy. IEEE Trans Pattern Anal Mach Intell 99(PrePrints):1
20. Arthur D, Vassilvitskii S (2007) K-means++: The advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '07, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. pp 1027–1035. http://dl.acm.org/citation.cfm?id=1283383.1283494
21. Powersmiths (2010) Powersmiths: Power for the Future. http://ww2.powersmiths.com/index.php?q=content/powersmiths/about-us
22. Bache K, Lichman M (2013) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
23. Hutchins J (2013) Freeway Performance Measurement System (PeMS). http://pems.dot.ca.gov/
24. Machine Learning Group (2012) Weka 3 - Data Mining with Open Source Machine Learning Software in Java. http://www.cs.waikato.ac.nz/ml/weka/