4-16-2014 12:00 AM

Contextual Anomaly Detection Framework for Big Sensor Data

Michael Hayes, *The University of Western Ontario*

Supervisor: Dr. Miriam A.M. Capretz, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

CONTEXTUAL ANOMALY DETECTION FRAMEWORK FOR BIG SENSOR DATA

(Thesis format: Monograph)

by

Michael A Hayes

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Engineering Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

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 thesis outlines a contextual anomaly detection framework for use in Big sensor Data systems. The framework 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. The contextual anomaly detection framework is evaluated with respect to two different Big sensor Data datasets; one for electrical sensors, and another for temperature sensors within a building. The results of the framework were positive in that we were able to detect content anomalies in real-time, and then pass these anomalies to the context detector. The context detector was able to reduce the number of false positives identified. Further, the approach is compared against the R outliers statistical package with positive results.

Keywords: Anomaly detection, point anomalies, context anomalies, Big sensor Data, predictive modelling, MapReduce

Acknowledgements

This thesis would not have been made possible without the support and guidance of my friends, family, colleagues, and many others that I have had the pleasure of working with in the past two years. I would first like to thank Dr. Miriam Capretz, Associate Professor in the Department of Electrical and Computer Engineering at Western University and my graduate supervisor for the past two years. I first began working with Dr. Capretz in the final summer of my undergraduate program and she has since been a driving force and motivator in my work. Her guidance, support, and expertise has been hugely important in shaping who I have become. Thank you for your hard work, patience, time, and effort you have given me these past several years.

To my mother, Margaret Fleming, and my father, Robert Hayes, who have given me love, appreciation, and seemingly infinite support throughout my life. You both have had instrumental roles in pushing me to become a better, more well-rounded person. To my brother, David Hayes, I thank you for friendship, motivation, and love. You are a more creative person than I could ever strive to be; I hope I made you proud, brother. Family: I sincerely dedicate this work to you, without all of your help I would not be where I am today.

Vanessa Rusu. You had to deal with a lot these past few years. I thank you for your patience and love, especially through the late nights. You have seen me in my lowest and highest moments, and for that I am deeply grateful. Thank you Vanessa, you were a rock of support for me, and I do not know how I could have possibly done this without you.

I would also like to thank all of the many colleagues I have had the privilege of working with these past two years. David Allison, Kevin Brown, Katarina Grolinger, Wilson Higashino, Abhinav Tiwari, and Alexandra L'Heureux: you all deserve immense credit for helping shape my ideas from random thoughts to directed solutions. I have also had the priviledge of working with many professors, and industry, that have helped shape this thesis and myself as a researcher. To the professors I have had the opportunity to teach with, thank you for having me as a teaching assistant. I hope some of your charm and talent as an educator rubbed off on me. To the members of industry I worked with, thank you for your support and healthy discussions on the application of research in real-world problems.

I would finally like to thank my grandparents, extended family, friends, and others I have missed. I have been fortunate to have some wonderful people in my life, and I thank you for your support.

Contents

List of Figures

List of Tables

List of Listings

List of Appendices

Chapter 1

Introduction

Anomaly detection is the identification of abnormal events or patterns that do not conform to expected events or patterns[15]. Detecting anomalies is 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[15]. 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.

Anomaly detection can be divided into detection of three main *types* of anomalies: point, collective, and contextual. Point anomalies occur when records are anomalous with respect to all other records in the dataset. This is the most common type of anomaly and is well-explored in literature. Collective anomalies occur when a record is anomalous when considered with adjacent records. This is prevalent in time-series data where a set of consecutive records are anomalous compared to the rest of the dataset. The final type, contextual, occurs when a record is only considered anomalous when it is compared within the context of other meta-information. Most research focuses on the former two types of anomalies with less concern

for the meta-information. However, in domains such as streaming sensor applications, anomalies may occur with spatial, temporal, dimensional, or profile localities. Concretely, these are localities based on the location of the sensor, the time of the reading, the meta-information associated with the sensor, and the meta-information associated with groups of sensors. 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.

1.1 Motivation

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. Further, organizations that supply sensoring services compete in an industry that has seen huge growth in recent years. One area that these organizations can exploit to gain market share is through providing more insightful services beyond standard sensoring. These insightful services can include energy trends, anomaly detection, future prediction, and usage reduction approaches. All of this culminates in a future that includes smart buildings: that is, buildings that can manage, optimize, and reduce their own energy consumption, based on Big sensor Data [14].

The amount of data produced by buildings, organizations, and people in general, is continually growing. The existence and prevalence of Big Data offers the promise of building

intelligent decision making systems. This is because many decision making algorithms exploit the idea that more data to train the algorithm creates a more accurate model. A caveat is that this does not hold for algorithms that suffer from high bias [20]. However, algorithms that are highly biased can be improved upon through adding extra features; this is common in highly *wide* datasets, where there exists a large number of features. With all the benefits of Big Data and predictive modelling, there are certainly some drawbacks. One of the major constraints is in feature selection. As in most machine learning approaches, selecting the features which best represent the data model is a difficult task.

Along similar lines, another major constraint is in the types of algorithms used for predictive modelling in general, and thus anomaly detection as a subset of predictive modelling. Some consider that there is a paradigm shift in the types of algorithms used: from computationally expensive algorithms to computationally inexpensive algorithms. 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 [18]. 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 idea and shift it to incorporate these computationally expensive algorithms which still generally perform better. By applying a hierarchical approach which involves only calculating the expensive algorithm on a very small subset of the initial data, the approach can utilize the *best of both worlds*.

Real-time detection for Big Data systems is also a growing area of interest. One of the common attributes associated with Big Data is *velocity*. Velocity is not only the rate at which data is input to the system, but also the rate at which the data can be processed by the system. Ensuring algorithms can process data in real-time is not an easy task as many require huge computational overheads to calculate prediction scores. This is particularly important in fields

such as data visualization where real-time feedback by the user is important in skewing the algorithm to presenting appropriate visualizations. Data visualization is also a common method for determing anomalous values. Therefore, a motivation of this work is to present a framework that can cope with Big Data in real-time; while not neccessarily for data visualization, but for identification of real-time anomalies as they are streamed to the framework.

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[27], while others propose methods to define outliers based on relational class attributes [36]. 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[19]. 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 [45]. 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.

1.2 Thesis Contribution

There are a number of contributions for the work presented in this thesis; this section provides a summary of the main contributions. The primary contribution is to present a modular,

extendible, and scalable framework for contextual anomaly detection in Big sensor Data. We confine the contribution of this work to Big sensor Data; however, the framework is generalized such that any application which has *contextual* attributes and *behavioural* attributes can utilize the framework. An additional contribution is the ability to apply the framework to real-time detection of algorithms for use in Big sensor Data applications. We also stress that the framework presented is modular in its architecture, allowing the content and contextual detection algorithms be replaced by new research.

The second contribution is to provide a paradigm shift in the way algorithms are structured for Big Data. As research has progressed in the machine learning domain, and in particular the anomaly detection domain, algorithms become more and more computationally expensive, and accurate. Thus, another contribution of this work is to provide a hierarchical approach to algorithm construction for Big Data; where a computationally inexpensive, inaccurate, model is used to first determine initial anomalies. From this smaller pool of records, the computationally more expensive model can be used to reduce the number of false positives generated by the first model. This hierarchical model can be used in many other domains; here we present a framework for exposing this model specifically for Big sensor Data. The modularity presented in the first contribution also is tied into this hierarchical structure. Due to the separation of concerns existing within the framework, the modules can be independently created, merged, modified, and evaluated without effecting the other modules. This also means additional modules can be added to the framework, such as semantic detection algorithms, to further enhance the framework.

In summary, the thesis 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 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 contribution 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 results of the implementation for this research are positive. The framework is able to identify content anomalies in real-time for a real-world dataset and conditions. Further, the number of false positives identified by the content detector are reduced when passed to the context detector. Also, the framework has been compared against the R statistical outliers package with positive results: the anomalies identified by the framework are equivalent to those identified by R.

1.3 The Organization of the Thesis

The remainder of this thesis is organized as follows:

- Chapter 2 outlines both the background information associated with contextual anomaly detection for Big sensor Data, and a ltierature review of the current approaches to contextual anomaly detection for Big sensor Data. This chapter will first provide an introduction to common terminology that will be used throughout the rest of the thesis. Second, an

overview of practical approaches for anomaly detection will be explored, including an introduction to a variety of algorithms with their benefits and drawbacks. Finally, the chapter will provide a review of state of the art techniques presented in academia for contextual anomaly detection, and anomaly detection in Big Data.

- Chapter 3 contains the main contribution for this thesis. The chapter will provide a description of the components for the Contextual Anomaly Detection Framework (CADF). Included in the description is an overview of each individual components, and how they interact with one another. The chapter is broken into two components: the content anomaly detector and the context anomaly detector. The content anomaly detection section includes information on the algorithm used by the detector, the random positive detection component, and a complexity discussion to provide insight into the scalability for Big Data. The context anomaly detection section provides detail on generating the Sensor Profiles (SPs) using the MapReduce framework, the underlying context detection algorithm, and a complexity discussion. Finally, a summary of how the components interact with each other, and how their interaction provides modularity and scalability for Big Data is presented.

- Chapter 4 includes a description of the implementation and evaluation of the CADF. First, the implementation for each component of the CADF will be shown. In particular, a look at the types of datasets that can be evaluated by the implementation is presented. Finally, how the components will interact with each other in a virtualized sensor stream will be explored. This final discussion is important as the case studies that will be discussed in the evaluation section are offline datasets that are exported from real-world data streams. To be consistent with the real-world counterpart, the virtualized sensor stream was created. The second major section of this chapter is focused on the evaluation of the CADF implementation. This portion of the chapter involves describing the streaming sensor datasets for each CADF component and their results. Additionally, a comparison

with some offline anomaly detection algorithms is shown.

- Chapter 5 provides the conclusion for this work, as well as a discussion on possible future works for the CADF.

Chapter 2

Background and Literature Review

The goal of this chapter is two-fold: first, the concepts that will be explored in the proposed work will be discussed; second, an overview of the related research works for contextual anomaly detection will be presented. The literature review will include works specifically related to contextual detection in *Big Data*, but also for contextual detection for standard sized data.

2.1 Concept Introduction

This section will introduce concepts and terminology related to **anomaly detection, MapReduce, Big Data,** and **context**. These concepts will first be defined, and then discussed in the context of academia and implementation practice. Therefore, this section serves as a primer for the concepts that will be explored further in the rest of the thesis.

2.1.1 Anomaly Detection and Metrics

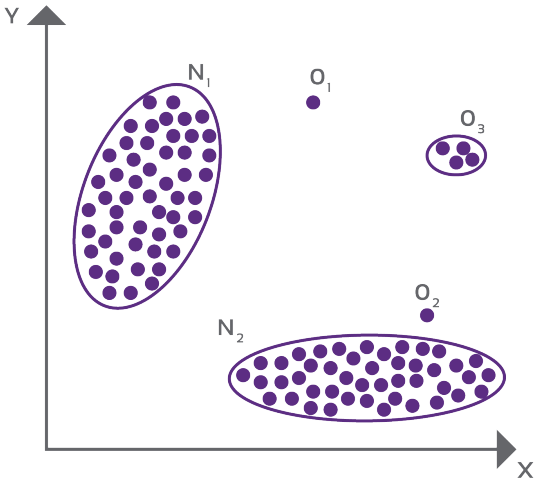
Anomaly detection can be categorized by three attributes: *input data, availability of data labels (anomalous versus normal)*, and *domain specific restraints* [34]. To begin, Table 2.1 introduces common terminology that is used to describe the first attribute: input data. One of the major

considerations in using an anomaly detection algorithm is based on the type of features present within the set of records, i.e. categorical, continuous, and binary. Further, another consideration are the relationships existing within the data itself. Many applications assume that there exist no relationships between the records; these are generally considered *point* anomaly scenarios. Other applications assume that relationships may exist; these are generally referreed to as *contextual* anomalies. The types of anomalies that may occur are discussed further in Table 2.2 [15].

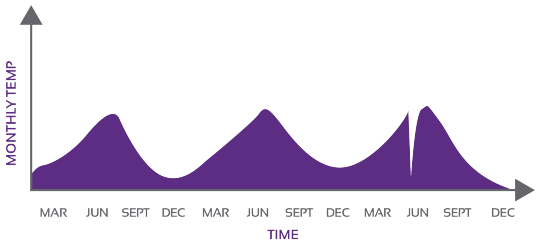
Term	**Definition**
Record	A data instance; for example: sensor data including the reading, location, and other information.
Feature	The set of attributes to define the record; for example: the reading, and location are each individual features.
Binary Feature	The feature can be of a binary number of possible values.
Categorical Feature	The feature can be of a categorical number of possible values; for example: a direction feature may be categorical with the set of values north, south, east and west.
Continuous Feature	The feature can be of a continuous number of possible values; for example: the sensor reading may be any floating point number [0, 100].
Univariate	The record is composed of a single feature.
Multivariate	The record is composed of several features.

Table 2.1: Anomaly Detection Definitions

Algoritms aimed at detecting anomalies can be categorized based on the types of data labels known apriori. Namely, if the algorithm knows a set of records is anomalous, it is referred to as a *supervised* learning algorithm. In contrary, if the algorithm has no notion of the labels of data, anomalous or otherwise, the algorithm is referred to as a *unsupervised* learning algorithm. Further, a third category titled *semi-supervised* learning algorithms, involves approaches that assume the training data only has labelled instances for the **normal** datatype. This is a more common approach to anomaly detection than *supervised* learning algorithms as it is norrmally difficult to wholly identify all the abnormal classes [25]. However, the most common approach to anomaly detection is unsupervised learning algorithms. The technique involved in these



(a) *Point Anomaly*: Indicated by O_1, O_2, and O_3, these values fall outside the otherwise dense regions of N_1 and N_2.



(b) *Contextual Anomaly*: Indicated by a drastic drop just before June, this value is not indicative of a normal value found during this time.

(c) *Collective Anomaly*: Indicated by the horizontal line midway through the graph. These values are all anomalous when considered contiguously against the normal pattern of the rest of the data.

Figure 2.1: Types of Anomalies

types of algorithms relies on the assumption that anomalies are far less frequent than normal records in the dataset. Further, an underlying assumption in most of these algorithms, regardless of category, is that they rely on the assumption that anomalies are dynamic in behavior. That is, it is very difficult to determine all the *labels* of anomalies for the entire dataset, and the future direction the dataset will take. Finally, most anomaly detection applications in practice have the anomalous and normal labels defined through human effort. As a result, it is much more common in practice to find datasets which are wholly unlabelled, or partially unlabelled, requiring the use of an unsupervised or semi-supervised learning algorithm [44].

This chapter will also introduce concepts related to some metrics used in the contribution and evaluation chapters: Chapter 3, and Chapter 4 respectively. The first concepts to introduce are related to *sensitivity* and *specificity* [31]. These are metrics used within classification tests to determine how well a classifier performs in classifying the correct versus incorrect errors. The definitions are presented in Table 2.3. While the subsequent chapters will not explicitly use

Term	Definition
Point Anomalies	A single record is considered to be abnormal with respect to the other records in the dataset. Point anomalies are the most common application for anomaly detection, and research in anomaly detection, as will be shown in Section 2.2. A web server identifies a user as a victim of identity theft based on anomalous web page visits. A pictorial example is shown in Figure 2.1a.
Contextual Anomalies	A record is anomalous within a specific context. For example, a sensor reading may only be considered anomalous when evaluated in the context of temporal and spatial information. A reading value of zero may be anomalous during normal working hours for a business, but non-anomalous during off-work hours. Pictorally this is shown in Figure 2.1b.
Collective Anomalies	A single record is considered collectively anomalous when considered with respect to the entire dataset, an example is shown in Figure 2.1c. Concretely, this means that the record may not be considered as anomalous alone; however, when combined within a collective of sequential records, it may be anomalous.

Table 2.2: Anomaly Detection Types Definitions

the definitions for specificity and sensitivity, they will make use of the terms that are used to define these terms; namely, **false positives, false negatives, true positives,** and **true negatives**. The term chart, Table 2.3, can also be explained more clearly through a figure, as shown in Figure 2.2.

2.1.2 MapReduce

MapReduce is a programming paradigm for processing large datasets in distributed environments [17] [19]. In the MapReduce paradigm, the *Map* function performs filtering and sorting operations, while the *Reduce* function carries out grouping and aggregation operations. The classic **Hello, World!** example for MapReduce is shown in Listing 2.1. In the example, the *Map* function splits the document into words, and produces a key-value pair for each word in the document. The *Reduce* function is responsible for aggregating the output of the *Map* functions. For each key, in this example **word**, the *Reduce* function evalutes the list of values,

Term	Definition
Sensitivity or *recall rate*	The proportion of actual positives identified as positive.
Specificity or *true negative rate*	The proportion of actual negatives identified as negative.
True Positive (*TP*)	Positive examples identified as positive.
True Negative (*TN*)	Negative examples identified as negative.
False Positive (*FP*)	Negative examples identified as positive.
False Negative (*FN*)	Positive examples identified as negative.

Table 2.3: Sensitivity and Specificity Definitions

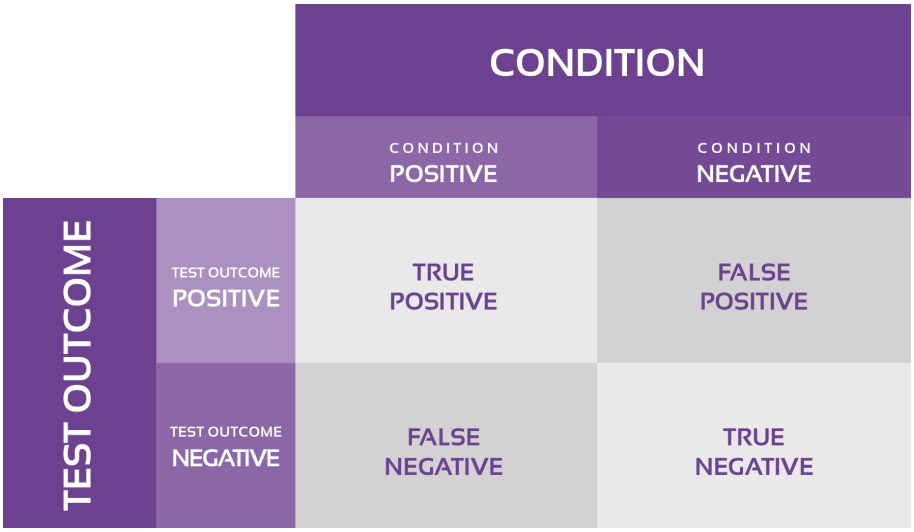


Figure 2.2: Sensitivity and specificity matrix defining the relationships between a test outcome and the expected condition.

in this example **partialCounts**. The *Reduce* function groups by word and sums the values received in the **partialCounts** list to calculate the occurrence for each word.

The main contribution of MapReduce paradigm is the scalability as it allows for highly parallelized and distributed execution over large number of nodes. In the MapReduce paradigm, the map or the reduce task is divided into a high number of jobs which are assigned to nodes in the network. Reliability is achieved by reassigning failed nodes job to another node. A well known open source MapReduce implementation is Hadoop which implements the MapReduce programming paradigm on top of the Hadoop Distributed File System (HDFS) [11], which is similar to the Google File System (GFS) [21]. A primary appeal of applying MapReduce for

machine learning algorithms, anomaly detection as an example, is to aid in scaling the machine learning algorithms to *Big Data*.

.

Listing 2.1: Code Snippet for a Simple Map and Reduce Operation

```
function map(name, doc) {
        for(word : doc)
                emit(word, 1);
}
function reduce(word, List partialCounts) {
        sum = 0;
        for(pc : partialCounts)
                sum += pc;
        emit(word, sum);
}
```

While there exists an appeal for MapReduce in scaling machine learning algorithms for *Big Data*, there also exist many issues. Specifically, when considering the volume component of Big Data, additional statistical and computational challenges are revealed [22]. Regardless of the paradigm used to develop the algorithms, an important determinant of the success of supervised machine learning techniques is the pre-processing of the data [35]. This step is often critical in order to obtain reliable and meaningful results. Data cleaning, normalization, feature extraction and selection are all essential in order to obtain an appropriate training set. This poses a massive challenge in the light of Big Data as the preprocessing of massive amounts of tuples is often not possible.

More specifically, in the domain of anomaly detection and predictive modelling, MapReduce has a major constraint which limits its usefulness when predicting highly correlated data. MapReduce works well in contexts where observations can be processed individually [38]. In this case the data can be split up, calculated, and then aggregated together. However, if there are

correlated observations that need to be processed together, MapReduce offers little benefit over non-distributed architectures. This is because it will be quite common that the observations that are correlated are found within disparate clusters, leading to large performance overheads for data communication. Use cases such as this are commonly found in predicting stock market fluctuations. To allow MapReduce to be used in these types of predictive modeling problems, there are a few potential solutions based on solutions from predictive modeling on traditional data sizes: data reduction, data aggregation, and sampling. Therefore, it is important to select aspects of the machine learning algorithm or framework to implement MapReduce efficiently and effectively.

2.1.3 Big Data

Big Data can be defined as massive and complex datasets composed of a variety of data structures, including structured, semi-structured, and unstructured data. Big Data was originally defined by the Gartner group as the three V's: volume, velocity, and variety [46]. In recent years, additional "V's" have been added to include: veracity, value, and visualization to cope with the evolving requirements of organizational benefits for addressing and understanding Big Data. Currently, businesses are acutely aware that huge volumes of data, which continue to grow, can be used to generate new valuable business opportunities through Big Data processing and analysis [43]. Figure 2.3 illustrates the evolution of the three standard V's of Big Data.

Velocity refers to processing Big Data with speeds of real-time or near real-time. Velocity is also related to the rate of changes within the data; that is, data may be processed in bursts of entries, rather than at constant rates. This is important in many anomaly detection domains where the anomalies cause a reactionary effect. For example, in systems with complex event processing, the anomaly detection component can fire events for the complex event processor to react to. The reaction may be optimizing a cloud network that the complex event processor is responsible for, or the reaction may be modifying parameters in a building network to save

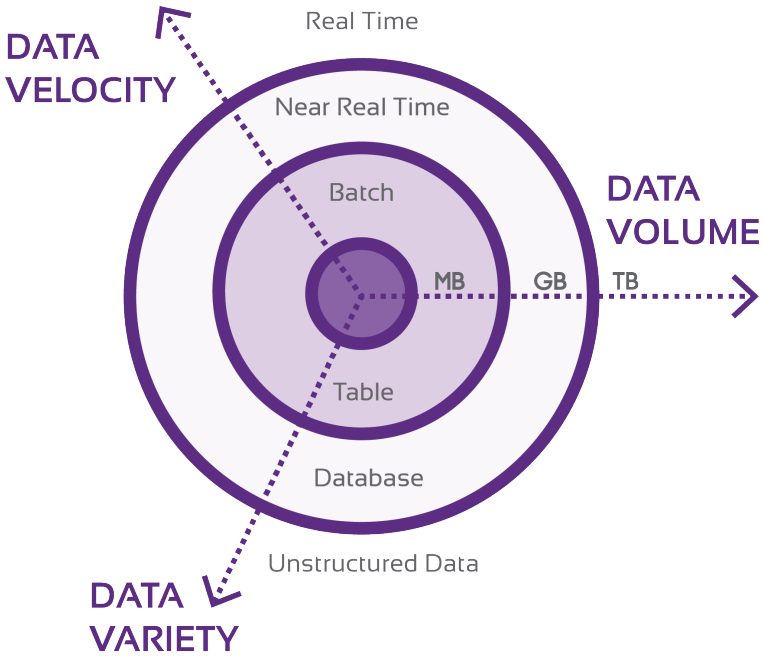


Figure 2.3: Classic 3 V's of Big Data: Velocity, Volume, and Variety.

energy [23].

Volume refers to the growing amount of data stored, from megabytes to pentabytes and beyond. The volume of data is important in the sensoring domain as normal operating parameters for sensor applications are thousands of sensors per location, each streaming sensor readings every minute. Within a company's storage system there may also exist hundreds or thousands of clients with several locations. This culminates in a huge volume of data being stored.

Variety refers to the component of Big Data that includes the requirement to store disparate datatypes such as: tables, other databases, photos, web data, audio, video, unstructured data, social media, and primitive datatypes such as integers, floats, and characters. Within the sensoring domain this is also important as sensors exist for a variety of sources, such as lighting, heating and ventillation, greenhouse gas emissions, hydro, and waste control. Further, sensoring companies also store audio, images, video, and demographic information related to their clients. Additionally, sensor networks may evolve overtime, resulting in a fluid data schema for the data storage system.

2.2 Anomaly Detection Techniques

In this section, various anomaly detection techniques will be presented that are generally used for detecting anomalies in streaming sensor networks. In particular, this section will describe the *pros* and *cons* of each approach in general. Further, this section will serve as a basis for why the algorithm for the contribution is selected in Chapter 3. There are some challenges for anomaly detection in sensor networks, namely, the prescence of noise in the data makes anomaly detection more challenging, and data is collected from distributed, disparate, sources. As a result, the approaches that will be discussed are:

- Bayesian Networks.
- Parametric Statistical Modelling.
- Nearest Neighbour-based Techniques.
- Rule-based Techniques.

2.2.1 Bayesian Networks

Bayesian networks are commonly used in multi-class anomaly detection scenarios. Bayesian networks are a classification-based approach whereby the network estimates the probability of observing a class label given a test record. The class label with the highest probability is chosen as the predicted class label for the test record. An example of a Bayesian network is shown in Figure 2.4. There are a few advantages to the classification-based Bayesian network approach. First, multi-class Bayesian networks use a powerful algorithm to distinguish between instances belonging to different classes. Second, the testing phase is fast. This is important in anomaly detection for sensor networks as real-time evaluation for *all* the sensor readings must be processed. Without a real-time testing phase, this is impossible.

Bayesian networks also have some disadvantages for anomaly detection that eliminate them from contextual-based approaches. First, multi-class classification, that is, where there are

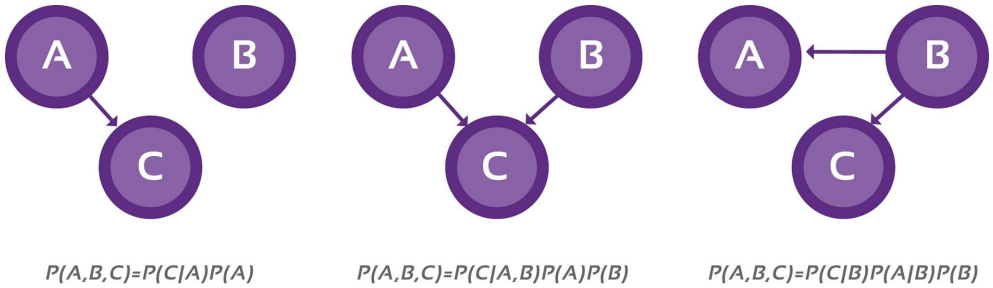


Figure 2.4: Bayesian Network Examples. The subcaptions indicate the formula to calculate the overall probability for each network.

multiple anomaly class labels or multiple normal class labels, rely on the readily available class labels for all the normal classes. As was previously discussed, this is generally not possible for practical purposes. Human intervention is normally required to label training records, which is not always available. Second, Bayesian approaches assign discrete labels to the test record. This is not disadvantageous for sensor networks as *meaningful* anomaly score is not always required. However, when evaluating the effectiveness of the anomaly detection approach, it is useful to discriminate between anomaly scores. This is more evident in *Big Data* contexts as a continuous anomaly score can aid in reducing the overall number of anomalies needing to be evaluted.

2.2.2 Nearest Neighbour-based Techniques

Nearest neighbour-based techniques rely on a single underlying assumption: normal data records occur in dense neighbourhoods, while anomalies occur far from their closest neighbours. The major consideration for nearest neighbour-based approaches is that a distance, or similarity, metric must be defined for comparing two data records. For some datasets this is simple; for example, continuous features can be evaluated with Euclidean distance. For categorical attributes, pairwise comparison of the feature can be used. For multiple features, each feature is compared individually and then aggregated. One of the main advantages of such an approach is that they are unsupervised. It was previously mentioned that a major considera-

tion for anomaly detection in Big sensor Data is that it is difficult to label the training data. For nearest neighbour-based techniques this is not necessary. Using nearest neighbour-based techniques is normally straightforward and only requires the definition of a similarity metric.

The definition of the similarity metric is one of the major drawbacks for nearest neighbour-based approaches. In many cases, including the Big sensor Data scenario, it is very difficult to determine a suitable similarity metric for the aggregation and variety of unstructured, semi-structured, and structured data. The performance of nearest neighbour-based approaches relies heavily on the similarity metric, and thus when it is difficult to determine the distance measure, the technique performs poorly. Further, as in all unsupervised techniques, if there is not enough similar *normal* data, then the nearest neighbour-based technique will miss anomalies. The biggest problem with nearest neighbour-based approaches is the computational complexity in evaluating the similarity metric for test records. Namely, the simliarity metric needs to be computed for the test record to all instances belonging to the training data to compute the nearest neighbours. This also needs to be done for *every* test data, an extremely difficult task for real-time evaluation of thousands of streaming sensor readings.

2.2.3 Statistical Modelling

Statistical modelling approaches rely on the assumption that normal records occur in high probability regions of a stochastic model, while anomalies occur in the low probability region. These techniques fit a statistical model to the given training dataset and then apply a statistical inference model to determine if the test record belongs to the model with high probability. There are two types of statistical modelling techniques: parametric and non-parametric models. Parametric techniques assume that normal data is generated by a parametric distribution, such an example is the Gaussian model and the Regression model. Non-parametric techniques make few assumptions about the underlying distribution of the data and include techniques such as histogram-based, and kernel-based approaches. There are several advantages to the statistical modelling technique to anomaly detection that make it work for Big sensor Data cases.

First, if assumptions hold true then the approach provides a statistically justified solution to the problem. Second, the anomaly score output by the statistical model can be associated with a confidence interval which can be manipulated by the algorithm. This provides additional information for the algorithm to use to improve the efficiency and performance of the algorithm. Finally, the Big sensor Data context normally follows a normal, or Gaussian, distribution.

However, there are several disadvantages to the statistical modelling approach. First, if the underlying assumption of the data being distributed from a particular distribution fails then the statistical technique will fail. Second, selecting the test hypothesis is not a straightforward task as there exists several types of test statistics. While these disadvantages certainly effect the Big sensor Data scenario, there are ways to overcome the disadvantages while retaining the aforementioned advantages. For example, evaluating the algorithm with respect to a variety of test statistics can be done apriori. Comparing the results of this step will allow the algorithm to selec the most appropriate test statistic for the given use case.

2.2.4 Rule-based Techniques

Rule-based techniques are quite similar in their disadvantages and advantages to the Bayesian network technique. Rule-based techniques focus on learning rules that capture the normal implementation of the system. When testing a new value, if there does not exist a rule to encompass the new value, it is considered anomalous. Rules can have associated confidence intervals which normalizes the proportion between the number of training cases evaluated correctly by the rule to the total number of training cases covered by the rule. The most obvious benefit of taking a rule-based approach is that the rules represent real-world reasonings behind why a value is anomalous. This is in stark constrast to some of the other approaches, such as statistical modelling, which provide parameters that mean little to a human observer. Further, like Bayesian networks, a rule-based approach shines since it only needs to compare itself with the existing rules, resulting in a fast testing time. This is especially true for rule-based approaches such as decision trees where optimization in traversing the tree can additionally be made. An

example of a decision-tree approach is shown in Figure 2.5.

However, like Bayesian networks, there exist some major disadvantages to the rule-based approach. First, rule-based approaches rely heavily on apriori knowledge of training records that are anomalous versus normal. In slight contrast to Bayesian networks, rule-based networks especially need a detailed set of training records which are considered as normal to ensure a sufficient set of rules are determined. Second, while it was previously mentioned that rule-based approaches are fast to test, this may not always be the case when there exists a large, complex, set of rules. Determining the number and complexity of the rules in a rule-based system is an important consideration. Too complex and the system performs with less efficiency; not enough complexity and the system provides a large number of false positives and false negatives.

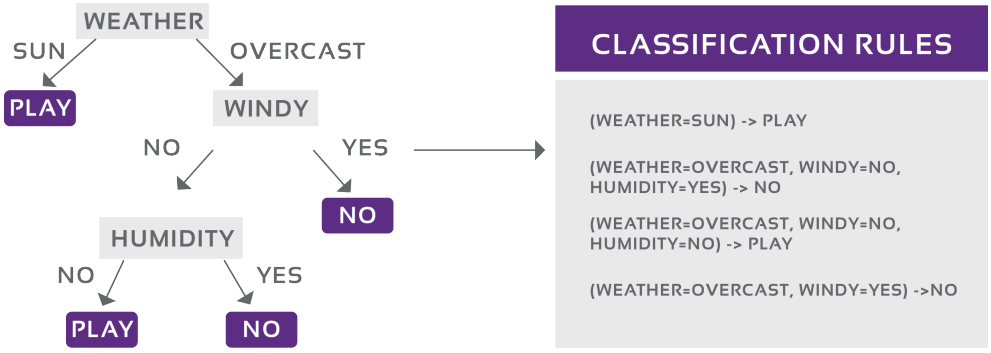


Figure 2.5: Decision-tree Rule-based Classification

2.2.5 Contextual Anomalies

Contextual anomalies are considered in applications where the dataset has a combination of contextual attributes and behaviour attributes. These terms are sometimes referred to as environmental and indicator attributes, as introduced by Song et al [48]. There are generally four common ways in which we define contextual attributes, these are defined in Table 2.4.

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

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 a company defines profiles for their users; should a new record violate the existing user profile, that record is declared anomalous.

Table 2.4: Definitions for Types of Contextual Attributes

attempts to apply separate point anomaly detection techniques to the same dataset, within different contexts. This requires a pre-processing step to determine what the contexts actually are, what the groupings of anomalies within the contexts are, and classifying new records as falling within one of these contexts. In the first 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. The most common approach to handling contextual anomalies is to reduce the problem into a point anomaly problem.

2.2.6 Parallel Machine Learning

When discussing anomaly detection algorithms in the context of Big Data it is important to note programming paradigms that have been developed to handle Big Data efficiently. Srinivas et al. [2] and Srirama et al. [49] both illustrate the importance and use of MapReduce as a programming paradigm for efficient, parallel, processing of Big Data. Efficient processing of Big Data is important because real-time predictive models are already well-established for smaller datasets. Therefore, future predictive models should conserve the ability to build real-time predictors when moving to larger volumes of data. Programming paradigms to achieve this have been seen in some popular software products such as Mahout [5], HaLoop [12], Spark [6], and Pregel [40]. These products attempt to scale machine learning algorithms (including predictive modeling techniques) to multi-machine architectures. However, there is still a research gap in addressing novel algorithms that are optimized specifically for predictive modeling in Big Data.

2.3 Anomaly Detection for Big Data

Anomaly detection algorithms in academia can be broadly categorized as point anomaly detection algorithms and context-aware anomaly detection algorithms[15]. 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[29]

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 [29] 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 thesis 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. [30] 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.

2.3.1 Predictive Modelling in Big Data

The prevalence and pervasiveness of Big Data offers a promise of building more intelligent decision making systems. This is because a consensus rule-of-thumb for many decision making algorithms is that more data can better teach the algorithms to produce more accurate outputs. One of the underlying concepts in these decision making algorithms is predictive modeling, or predictive analysis. That is, given a set of known inputs and outputs, can we predict an unknown output with some probability. Being able to construct an accurate prediction model is important in many disparate domains such as credit card fraud detection, user recommendation

systems, malicious URL identification, and many others. A common link between all these domains is their existing, and growing, amount of data. For example, companies such as Yahoo and Netflix collect a large variety of information on their clients to better recommend news or movies they predict the client will enjoy. The banking and credit industry collect demographic information as well as spending history to build reliable predictive models to determine loan valuations and fraudulent buying habits. Additionally, environmental companies collect energy usage data from sensors to define cost models, predict anomalous behaviour, and provide insight into reducing their client's carbon footprint.

One of the more controversial topics surrounding predictive modeling and Big Data is whether more data is really better for producing an accurate predictive model. Junque de Fortuny et al. [32] have recently produced some work regarding the usefulness of Big Data in the domain of sparsely filled datasets to construct accurate predictive models. The main contribution of their work is empirical proof that more data is indeed better when dealing with datasets such as Yahoo's movie database. Even more interesting, they show that collecting more fine-grained pieces of data provides a significant increase in the predictive model accuracy. An example they use is an online marketing company collecting web page visitation hits. The authors posit that by gathering even finer-grained information about client page visits, such as time spent on a page, will provide a better predictive model. Further, they propose that multiple companies, such as several small banks, should pool their data resources together to make more information available to the predictive model.

In contrast to the work of Junque de Fortuny et al. are interviews and presentations at the Predictive Analytics World Conference that warn against some of the promises of Big Data and their use in predictive modeling [13]. Specifically, Michael Berry, the analytics director at TripAdvisor, notes that predictive models succeed primarily because of the features chosen for building the model, not the amount of data delivered to generate the model. Further, Berry suggests that finding patterns in datasets generally occurs rather quickly, so an increase in the amount of data is only an increase in the time it takes for the predictive model to converge, not

in the patterns that are found. Other professionals at the conference, such as Eric Feinberg, senior director of mobile, media and entertainment at ForeSee, offer that the benefits of Big Data predictive modeling vary depending on the industry in question. For example, determining outliers or anomalies in data is easier, and in some cases only possible, when there is ”enough” data. Otherwise, the anomalies may just been seen as noise and ignored. The latter scenario is useful in recommendation systems as the noise can normally be safely ignored. However, in an industry such as fraud detection, the anomalies are the target of the predictive model, and so Big Data is appropriate and necessary to build an accurate model. The ambiguity amongst researchers as to the benefits of Big Data in predictive modeling leads to questions of its usability and usefulness. One open research area is to conduct a formal analysis on the benefits that predictive modeling can achieve with Big Data; or at least, to identify the industries that can benefit from Big Data generated predictive models.

The concept of noise has also provided a paradigm shift in the types of machine learning algorithm used for predictive modeling in Big Data. Dalessandro [18] illustrate the usefulness of accepting noise as a given, and then using more efficient, but less accurate, learning models. Dalessando shows that using algorithms such as stochastic gradient descent, a less complex optimization technique that provides a higher optimization error, with a large amount of data actually provides better estimation and approximation error, at a much faster rate than the more computationally complex algorithms. Concretely, this means that the algorithm defines a model which performs better in estimating a new output, even though it may underperform compared to other algorithms during the optimization calculation step. Without Big Data, inferior algorithms such as stochastic gradient descent will not only perform worse in optimization error but also in prediction and approximation error. However, when used with a large amount of data that can train the algorithm over millions of iterations, a faster, and more (or equally) accurate predictive model will be constructed [52]. This shift from more complex to less complex optimization techniques leads to an open problem in predictive modeling: how do we build a scalable predictive model using dated optimization models, but with new programming

paradigms such as MapReduce? Due to this, there is also room for novel predictive modeling algorithms that can exploit this knowledge to achieve better efficiency and accuracy.

The work presented in this thesis is applied to a case study for streaming sensor data. There are numerous other applications involving Big Data and predictive modelling. Ayhan et al. [8] present work involving predictive modelling, in general, for Big Data found in aviation. Their work illustrates an entire predictive modelling system for query processing and analytics for streams of aviation Big Data in real-time, or as some results show, near real-time. The work presented by Ayhan et al. involves performing data warehousing-type analytics over the data. That is, using techniques such as ”slice-and-dice” where dimensions for the data are pre-selected, and these dimensions can be selected together to provide subsets of useful information. Their work is specifically tailored to the underlying data structure of the aviation data their research is applied to and is thus difficult to expand to other domains.

2.3.2 Real-time Analytics in Big Data

Interactive analytics can be defined as a set of approaches to allow data scientists to explore data in an iterative way, supporting exploration at the rate of human thought [28]. Interactive analytics on Big Data provides some exciting research areas and unique problems. Most notably, and similar to other data analytic approaches, is the question how can we build scalable systems that query and visualize data at interactive rates? The important difference to other data analytic paradigms is the notion of interactive rates. By definition, interactive analysis requires the user to continually tweak or modify their approach to generate interesting analytics.

A large category of interactive analytics is data visualization. There are two primary problems associated with Big Data visualization. First, many instances of Big Data involve datasets with large amount of features, wide datasets, and building a highly multi-dimensional visualization is a difficult task. Second, as data grows larger vertically, tall datasets, uninformative visualizations are generally produced. For these tall datasets, the resolution of the data must be limited, i.e. through a process such as binning, to ensure that highly dense data can still be

deciphered [28]. The generic solution to this problem is that the scalability of data visualization should be limited by data resolution, not the number of rows. For highly wide datasets, a preprocessing step to reduce the dimensionality is needed. Unfortunately this tends to be useful on tens to hundreds of dimensions, for even higher dimensions a mixed-initiative method to determine subsets of related dimensions is required [28]. This approach generally requires human input to determine an initial subset of ”interesting” features, which is also a difficult task and open research area.

One approach to addressing issues associated with Big Data interactive analysis is through iterative processing. Applications such as Stat! [9] deploy an interactive analytics workbench to iteratively refine and test queries in short latencies. Applications such as Stat! rely on the idea that the user can receive partial information and then act, or refine, their query appropriately, known as progressive analytics [16]. Concretely, progressive analytics give quicker, less accurate, feedback on fewer resources, that progressively become more refined. A major research area for progressive analytics is in the sampling algorithm. As many Big Data case studies involve sparse datasets, it is normally difficult to sample an appropriate subset of the data.

A common component in machine learning approaches to Big Data is parallelism. Programming paradigms such as MapReduce have been discussed as one such parallelism model for interactive analytics [10]. Another approach specifically for interactive analysis is Google’s Dremel system [41], which acts in complement to MapReduce. Dremel builds on a novel columnar storage format, as well as algorithms that construct the columns and reassemble the original data. Some highlights of the Dremel system are:

- Real-time interactivity for scan-based queries.
- Near linear scalability in the number of clusters.
- Early termination, similar to progressive analytics, to provide speed tradeoffs for accuracy.

Other interactive analytics research has been based on the columnar data storage approach [1] [24]. The main benefit of column-based approaches versus row-based, traditional, approaches is that only a fraction of the data needs to be accessed when processing typical queries [24].

2.3.3 Computational Complexity

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 [45]. This is difficult in scenarios such as environmental sensor networks where there is little to no labelling for each sensor value. Shilton et al.[47] propose an 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. This is a drawback to their approach as it is generally difficult to entirely label values as anomalous apriori. 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. Parameter selection is a common problem amongst many machine learning algorithms; it is not specific to predictive modelling.

To reduce the effect of the computational complexity of these algorithms, Lee et al.[37] have proposed work to detect anomalies by leveraging Hadoop. Hadoop is an open-source software framework that supports applications to run on distributed machines [4]. 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[51]. 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.

2.4 Contextual Anomaly Detection

Little work has been performed in providing context-aware anomaly detection algorithms. Srivastava and Srivastava [50], 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. With all the added complexity of their approach, it is difficult to apply their work in a real-time application. However, concepts that are applied are somewhat similar to the work presented in this thesis. That is, combining a computationally expensive step to provide detailed insight into the detected anomalies with a computationally less expensive step to detect the initial anomalies.

2.4.1 Post-Processing Detection

Mahapatra et al.[39] 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. Mahapatra et al.'s [39] work is in contrast to the work proposed in this thesis as we are concerned with contextual information such as spatio-temporal relationships between sensors. Similar to the work proposed in this thesis is their use of contextual detection as a post-processing step. This allows the algorithm to be com-

pared and optimized at two distinct steps: point anomaly detection, and contextual anomaly detection. This is perhaps the major other work that presents a hierarchical approach to contextual anomaly detection. However, they present their work in this way to exploit the use of well-defined semantic libraries that exist independently. The research presented in this thesis aims at delivering a hierarchical approach to streamline the real-time processing for contextual anomaly detection, allowing extension and use of the proposed framework for Big Data applications.

Miller et al.[42] 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[42] 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. In contrast to the work proposed in this thesis, as the thesis will show a simulation of real-world datasets being processed by the algorithm in real-time using the content anomaly detector. Another dissimilarity is that Miller et al.'s approach requires the data to reside in some graph structure. This is generally not the case, especially in a dynamic environment such as a streaming sensor network. This constraint focuses their work on a small subset of applications where as the work presented in this thesis can be applied to any Big Data application where context and content coexist within the application's schema; that is, through spatial, temporal, or other meta-informational features.

2.4.2 Hierarchical-based Approaches

The work presented in this thesis relies of 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. [33] 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 thesis. Concretely, Kittler et al. [33] present context as more analgous to *semantics* in that the additional information they add is similar to domain ontologies rather than contextual information inherently associated within the data.

A different approach for contextual detection is that work of AlEroud et al. [3], who apply contextual anomaly detection to uncover zero-day cyber attacks. Their work involves two distinct steps, similar to the modules described in this thesis: 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. [3] is similar to the work presented in this thesis 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.

2.5 Summary

In this chapter an introduction to the concepts and techniques involved in anomaly detection for Big sensor Data was presented. Specifically, a discussion on the terminology and background related to anomaly detection, Big Data, and MapReduce, a common framework for implementing machine learning algorithms for Big Data, was presented. An overview of the different common techniques to perform anomaly detection in practice was provided, specifically in the context of Big sensor Data. Finally, a presentation of the academic works in the domain of anomaly detection, and contextual anomaly detection, in Big sensor Data was provided. The findings of the academic works were shown in addition to the research gaps that remain among the existing research. The research gaps that were discussed will be addressed in the contribution of this work, provided in Chapter 3.

Chapter 3

Contextual Anomaly Detection Framework

3.1 Contextual Detection for Big Sensor Data

The proposed contextual detection framework for use in Big Sensor Data is based on using a hierarchical approach to determine the anomalies that exist based on content alone, and then to determine anomalies that exist based on their contextual meta-information. Therefore, this chapter will outline how the content anomaly detector determines sensor readings which are anomalous and how the contextual anomaly detector determines anomalous readings. Further, this chapter will describe the MapReduce approach to building the clustering parameters which determine the Sensor Profiles (SPs) used in the contextual detector. Finally, how these separate detectors interact will be presented.

3.1.1 Hierarchical Approach

One of the major considerations for the contextual anomaly detection framework is scalability for Big Data. To accomplish this, the framework uses a hierarchical approach in applying a computationally inexpensive algorithm over *every* new data point, and a computationally

expensive algorithm over specific data points labelled as anomalous by the first algorithm. This dual relationship is outlined in Figure 3.1. In particular, this approach overcomes many of the drawbacks for traditional approaches in Big Data; namely, scaling computationally expensive linear algebra, and maintaining prediction accuracy with low prediction error. This approach holds for the framework proposed in this thesis; further, this thought paradigm can be applied to many other case studies, allowing new algorithms to exist efficiently in Big Data applications.

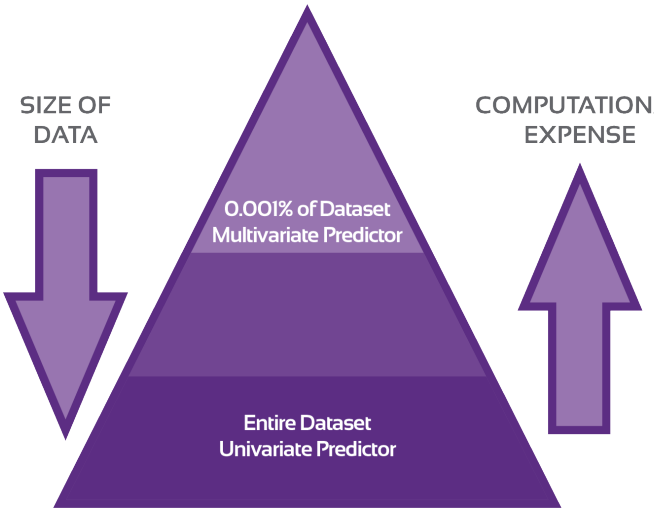


Figure 3.1: Contextual Anomaly Detection Framework Hierarchy

3.1.2 Contextual Detection Overview

An overview of the contextual detection framework, henceforth referred to as CADF, is shown in Figure 3.2. The framework is broken into several components, denoted by boxes: **sensors, content detection, context detection,** and **build clusters**. The components are connected by unidirectional and bidirectional arrows, indicating the direction of component-component communication within the framework. For example, the sensors are unidirectional and feed information directly to the first hierarchy, the content detection. However, the relationship link between the context detection and build clusters component is bidirectional. The bidirectional relationship is because the build clusters component learns the SP groupings, but also continually relearns and updates the SP groupings when an anomaly is detected. More discussion

on each of these components will occur in the coming subsections. A final important note regarding the CADF overview is that the sensors are not only connected to the content detection component, but also to a centralized database. This database is used by the **build clusters** component to train the clustering algorithm for determining the SPs, offline. The database is also used to train parameters used by both the **content detector** and **context detector**.

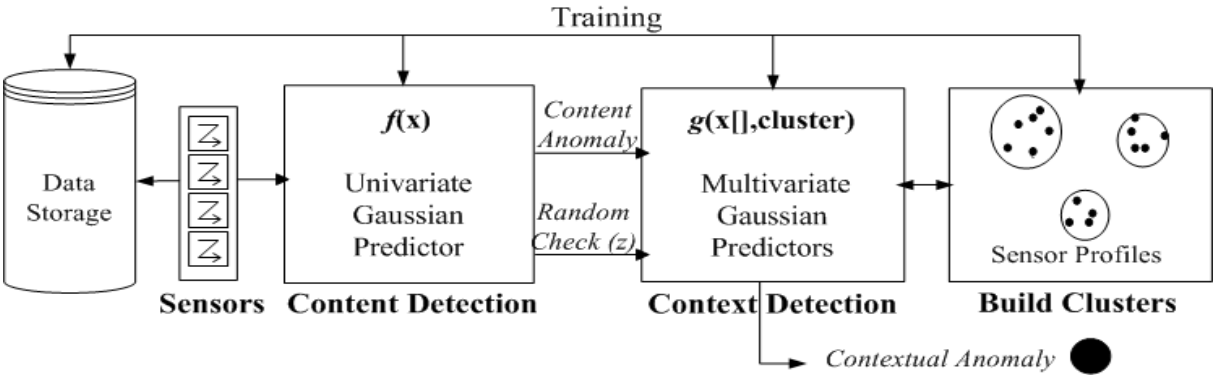


Figure 3.2: Contextual Anomaly Detection Framework

Algorithm 1 provides a description of the CADF with respect to function calls and is used to further define the workflow of the CADF. The functions defined in Algorithm 1 map to those components in Figure 3.2. In particular, `UnivariateGaussianPredictor` corresponds to the **content detection** component; `GetSensorProfile` corresponds to the **build clusters** component; and `MultivariateGaussianPredictor` corresponds to the **context detection** component. Further, the two incarnations of the `IsAnomalous` function compare the predicted value within a margin of error to determine if the current reading is anomalous. Section 3.2 and Section 3.3 will go into further detail for the content detector and context detector respectively.

The CADF is based on two important principles. First, the components within the CADF are functionally independent from one another. Second, the components witin the CADF increase in computational complexity while dealing with a decreasing amount of data. These two principles allow for the framework to easily expand for future work in anomaly detection, providing easy integration for new detection algorithms. The principles also provide scalability for use in the context of Big Data. Concretely, the separation of concerns between the context

Algorithm 1: Contextual Anomaly Detection

```
input  : SensorValue, SlidingWindow, SensorHistory
output: Anomaly
content ← UnivariateGaussianPredictor(SensorValue, SlidingWindow,
SensorHistory)
if IsAnomalous (content) || IsRandomContextCheck (content) 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
```

and content detectors reduces the number of computations the more computationally expensive context detector needs to perform. The relationship between the context and content detectors is analagous to catching fish in a large lake, or ocean. The content detector uses a wide net that may capture the target fish, and also a large amount of fish that may be similar, but not matching the target. The context detector uses a much smaller, more accurate, net to sift through the smaller pool of fish to determine the correct targets.

3.2 Content Anomaly Detection

The component which faces the raw data, i.e. sensors or some other real-time data stream, is the content detector. As previously mentioned in Section 3.1.2, the content detector is responsible for detecting anomalies in real-time to pass to the context detector. As a result, the content detector is computationally inexpensive, as it may need to be evaluated on hundreds, or thousands, or tens of thousands, of values every second, or minute. Due to this constraint, and

knowing that any values detected as anomalous are still evaluted using the context detector, the content detector is much more inaccurate. This was also mentioned in Section 3.1.2 and outlined in the fish-catching analogy. To reiterate, the content detector sacrifices accuracy for efficiency, because any inaccuracies will be remediated by the contextual detector.

There are several potential underlying techniques for designing the content detector in a streaming sensor context: Bayesian networks, rule-based systems, parametric statistics, and nearest-neighbour. Section 2 describes these techniques in detail, and also describes why the **Univariate Gaussian Predictor** is selected. To summarize, the univariate Gaussian predictor allows for the definition of a confidence interval. The CADF can make use of this confidence interval to trade-off accuracy for more efficient computation. Univariate Gaussian approaches also allow for unsupervised learning, which is important when dealing with a sensor network that may have anomalies which cannot be easily pre-defined. The major disadvantage for univariate Gaussian models is their reliance on the data model being generated from a Gaussian distribution. This generally holds true for streaming sensor networks, as the distribution assumption normally fails under highly dimensional data. However, we are only interested in a single dimension: the sensor value itself.

3.2.1 Univariate Gaussian Predictor

The Univariate Gaussian Predictor relies on learning two parameters during the training of the algorithm: μ and σ. Equation 3.1 and Equation 3.2 show how these two parameters are learned, where m is the number of training examples, and $x^{(i)}$ is the sensor reading for a training value, i. From Algorithm 1, the `IsAnomalous` function determines whether the predicted value from the univariate Gaussian predictor, $p(x)$, is less than some threshold, ϵ, where ϵ is set during implementation. Tweaking ϵ determines the confidence interval for how close the predicted value needs to be, compared to the Gaussian model. Increasing ϵ allows for more anomalies to be detected, essentially widening the net used in the running analogy for this chapter. Equation 3.3 defines the equation which will be evaluated in real-time for each new sensor reading. Ini-

tially, this equation begins as a sequence of products for each dimension of the data. However, the content detector is only considering the sensor value itself, and so only needs to calculate the product sequence once. Thus, the content detector needs to calculate a single exponential for each incoming sensor reading. As the number of concurrent sensors increases, so does the computational complexity of the content detector. However, comparatively this increase is small against the computational complexity of the context detector.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \tag{3.1}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2 \tag{3.2}$$

$$\begin{aligned} p(\bar{x}) &= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{(\bar{x}_j - \mu_j)^2}{2\sigma^2} \\ &= \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{(\bar{x} - \mu)^2}{2\sigma^2} (\because n = 1) \end{aligned} \tag{3.3}$$

3.2.2 Random Positive Detection

A component that was not previously discussed from Figure 3.2 is the random positive detection aspect of the content detector; this is shown as `IsRandomContextCheck` in Algorithm 1. In Figure 3.2, this is shown by a unidirectional arrow from the **content detector** component to the **context detector** component. One possible scenario is that the content detector may not be able to detect a sensor which is acting *normal* within its own sensor context; however, when evaluated with respect to other contextually similar sensors, the value is anomalous. This scenario can occur when a sensor is providing readings within the bounds of its maximum and minimum values, however, it may be producing these values during an off-work day (i.e. Saturday in a building with working hours of 8:00-19:00 Monday-Friday). This contextual information is not present for the content detector as it is only concerned with the reading;

however, the contextual detector will flag this as an anomaly. Therefore, there should be some mechanism to *randomly* send values which pass content anomaly detection to the contextual detector. Concretely, the random positive detection mechanism is used to reduce the number of *false negatives* produced by the CADF. From Section 2, a *false negative* is a test result that is incorrect because the test failed to recognize a *correct* finding. In the CADF, the random positive detection mechanism is an independent probability calculated for every real-time sensor reading. The probability, z is determined during implementation and can be modified depending on the case study.

3.2.3 Complexity Discussion

The major property of the content detector is its computationally efficiency. The algorithm can be inaccurate in its detection of *false positives*, as long as it is detecting the majority, or all, of the *true positives*. From Section 2, this means we want the algorithm to recognize correct findings as correct, even if we also detect incorrect results as correct. However, the algorithm cannot take a lot of wall clock time to do this because the content detector may be evaluated hundreds, to tens of thousands of times per second. If the detector even requires hundreds of milliseconds to perform the evaluation, then it cannot be used in the Big Data context. As mentioned in Section 3.2.1, and shown in Equation 3.3, the content detector requires a few multiplications, divisions, squares, and one exponential. Concretely, this means that the computational complexity of the algorithm is $O(n^{\frac{1}{2}}M(n))$ where n is the number of precision required by the case study and M(n) is the complexity of the multiplication algorithm. This assumes the algorithm uses repeated argument reduction and direct summation for evaluating the Taylor series.

3.3 Context Anomaly Detection

The context anomaly detector component is used after the content detector determines a sensor reading is anomalous, or for random sensor readings. The context anomaly detector component is composed of two sub-components: defining the *sensor profiles*, SPs, and assigning each sensor to one of the sensor profiles; and evaluating sensor values against the corresponding sensor profile detector model. These two components are shown in Figure 3.2 as **context detector** and **build clusters**, corresponding to evaluating the context detector and generating the sensor profiles respectively. The context anomaly detector is different than the content anomaly detector in a few major ways: first, it incorporates the contextual meta-information associated with the sensors. Second, it utilizes a Multivariate Gaussian Predictor to include the interdependencies between the contextual meta-information in addition to the sensor value. Finally, the context detector uses sensor profiles to enhance the context information, reducing the number of *false positives* generated by the content detector.

Table 3.1 defines meta-information that can potentially be associated with a sensor. Section 4.2 will describe an evaluation and implementation scenario in more detail, including a real-world scenario with real meta-information. Table 3.1 outlines some possible contexts that can be extracted from common features. For example, a **time** feature which is represented by DD/MM/YY HH:MM can have context such as *Day of the Week*, *Time of Day*, *Holiday*, or *Season* extracted from it. Additionally, a sensors spatial information can be extracted into *Floor, Sector, Company* (when the dataset includes multiple clients for a single provider), and *Room Usage* (such as research and development, office, or utility).

3.3.1 Sensor Profiles

The sensor profiles, SPs, in the CADF are contextually similar sensors that are clustered into groups that share characteristics. For example, when building SPs for a company building a streaming sensor network, different clusters may form to differentiate between heating and

Feature	Domain	Context
Time	DD/MM/YYYY HH:MM	Day of the Week
Time	DD/MM/YYYY HH:MM	Time of Day
Time	DD/MM/YYYY HH:MM	Holiday
Time	DD/MM/YYYY HH:MM	Season
Sensor Location	[a-zA-Z0-9]	Floor
Sensor Location	[a-zA-Z0-9]	Sector
Sensor Location	[a-zA-Z0-9]	Company
Sensor Location	[a-zA-Z0-9]	Room Usage

Table 3.1: Context Examples

ventillation sensors, electricity sensors, and hydro sensors. Additionally, clusters may form for even higher dimensional relationships such as floor levels, building zones, and operating hours. The SPs are then used by the context detector to build *separate* multivariate Gaussian predictors for each profile. Therefore, a new sensor value can determine which cluster it belongs to and evaluate itself with respect to the sensor profile's Gaussian predictor.

A two-dimensional representation of the clustering process for a sensor network is shown in Figure 3.3 where each circle represents a sensor and its meta-information. In the two-dimensional example, the dimensions may be electricity reading range and operating hours. The clustering algorithm could then determine three groups of sensors which operate under similar circumstances, combining them into three SPs. Defining SPs is important because it allows the CADF to scale to Big Data as sensors need to only evaluate themselves within a small context of similar sensors, instead of in the context of the entire sensor network. In this way, more sensors can be added to the network and have their sensor profile determined, maintaining the total number of clusters in the network, thus maintaining the scale of the application.

3.3.2 Clustering with MapReduce

One of the underlying goals of this research is to provide a scalable solution for anomaly detection in *Big Data*. A major bottleneck in the proposed framework is the clustering algorithm. While the clustering algorithm will be performed offline, it is still important to attempt to opti-

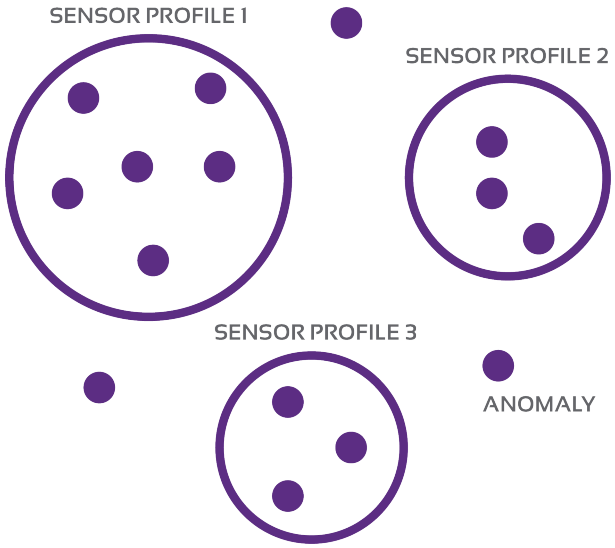


Figure 3.3: Sensor Profile Definition

mize the process, such that the clustering algorithm can be re-learned and updated frequently. In recent years, a common way to migrate existing serial solutions to Big Data is a parallelism model, such as MapReduce. In this section, we will define such a MapReduce approach to parallelizing the clustering algorithm for use in CADF. In particular, we will present a MapReduce approach for the common k-means clustering algorithm. K-means clustering aims to partition the dataset into a set of clusters that minimize the sum of squares distance between each data point. To do this, 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 3.4; 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.

$$\min_{s} \sum_{i=1}^{k} \sum_{x_j} ||x_j - \mu_i||^2 \tag{3.4}$$

From Chapter 2, 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 3.4 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.

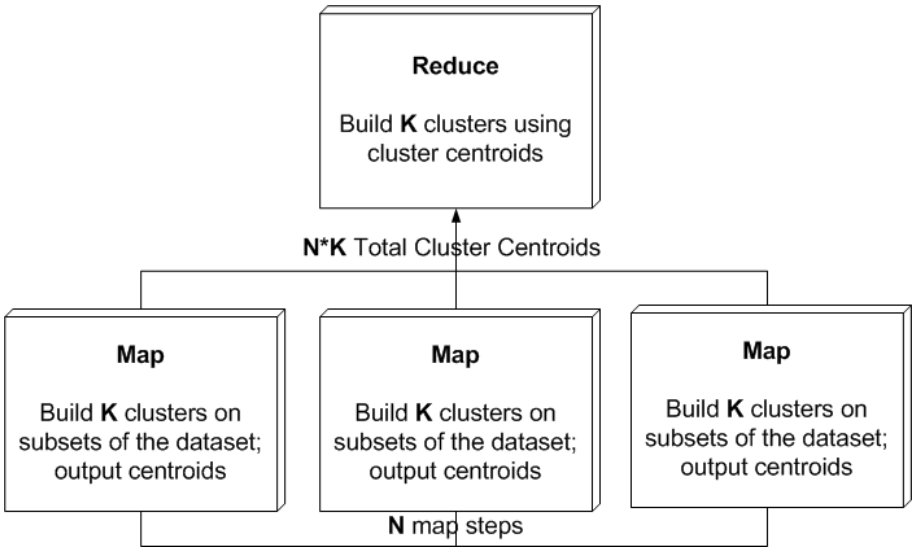


Figure 3.4: MapReduce to Determine Global Sensor Labels

The Map() and Reduce() steps in Figure 3.5 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.

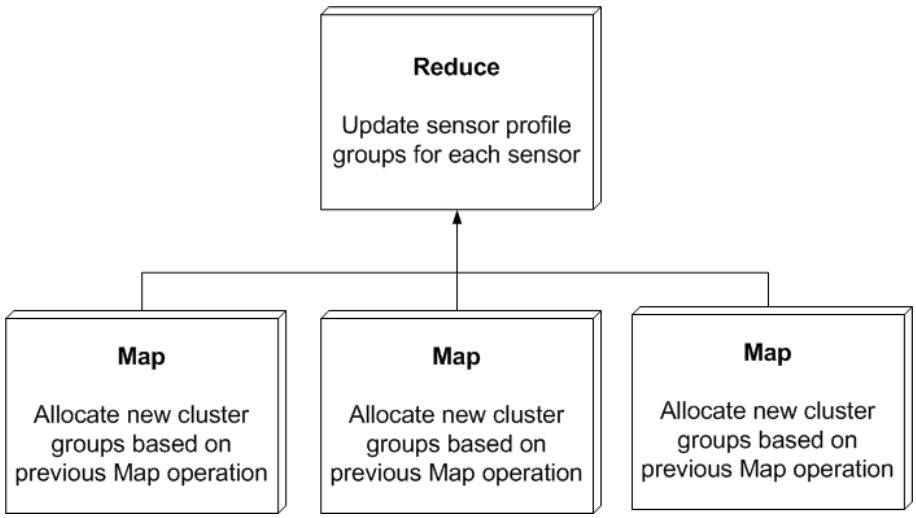


Figure 3.5: MapReduce to Re-label Sensors to Global Sensor Profiles

3.3.3 Multivariate Gaussian Predictor

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 3.5, 3.6, and 3.7 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 3.7. The value, *p(x)* is compared against some threshold, ϵ, and is flagged as anomalous if it is less than ϵ. Utilizing a multivariate Gaussian predictor allows the algorithm to inherently determine any interdependencies that are not wholly obvious to the observer. The covariance matrix provides a weighted view for the dimensions of the meta-information associated with each sensor within the corresponding sensor profile. Then, when a new value is entered, the meta-information of the new value's sensor is also included in the prediction calculation.

There are similar arguments for selecting the multivariate Gaussian predictor as the under-

lying algorithm of choice for the contextual detector as there were for selecting the univariate Gaussian predictor for the content detector. Most importantly, the multivariate Gaussian predictor has a property which inherently determines inter-relationships within the meta-information for the sensors. A common difficulty in any machine learning algorithm is feature selection. This difficulty is compounded for anomaly detection where it is difficult to know which features contributes to anomalies, which occur for less than 0.001% of the values. Therefore, the choice of algorithm comes down to selecting a statistical approach, like the multivariate Gaussian model, or a neural network approach which learns intermediate relationships with their hidden layers. There are a few major problems in choosing a neural network, which were discussed in more detail in Chapter 2. To summarize, a neural network requires a larger pool of deterministic training samples, whereas a Gaussian predictor performs well under unsupervised case studies.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \tag{3.5}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T \tag{3.6}$$

$$p(\bar{x}) = \frac{1}{(2\pi)^{(\frac{n}{2})}|\Sigma|^{(\frac{1}{2})}} \exp\left(-\frac{1}{2}(\bar{x} - \mu)^T\Sigma^{-1}(\bar{x} - \mu)\right) \tag{3.7}$$

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.

3.3.4 Complexity 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.

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.

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 [7] 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.

3.4 Summary

The context anomaly detection framework provides a modular, hierarchical, scalable method to detecting anomalies in *Big Data* scenarios. In particular, the modular approach allows for future expansion and reusability of the components by replacing the underlying context or content detector algorithms with future research. This may prove to be more important as a shift in the thought paradigm of dealing with Big Data has shifted from developing computationally expensive algorithms over small amounts of data to developing computationally inexpensive algorithms for large amounts of data. Even though the computationally inexpensive algorithm may have a larger intermediate estimation error, this error is slowly reduced as more data is used.

The hierarchical nature of the proposed CADF provides a way for the framework to scale to *Big Data*. The first tier of the hiearchy is evaluated for every new sensor value flowing through the system, while the second tier computes a computationally expensive anomaly detection algorithm to determine true anomalies. Concretely, this means that the algorithm's first tier identifies a large number of *true positives* and *false positives* while the second tier reduces the number of *false positives*. Additionally, since the first tier is computationally inexpensive, the algorithm can scale for Big Data while still maintaining its accuracy. The accuracy is maintained since the computationally more expensive, more accurate, algorithm is only evaluated on a small percentage of the total number of sensors, likely less than 0.001%.

Finally, the CADF incorporates aspects such as the random true detection, and parallelism such as MapReduce, to ensure the framework can overcome some of its drawbacks. Namely, the framework's most expensive aspect: training the clustering algorithm can be minimized by

utilizing a novel MapReduce algorithm to determine the cluster centroids in parallel. Also, to reduce the number of *false negatives*, the random true detection component will pass sensor values that passed content detection to the context detector. This will help identify sensor values that are *normal* with respect to its historical sensor values, but *abnormal* with respect to contextually similar sensors, i.e. within the sensor's *sensor profile*.

Chapter 4

Contextual Anomaly Detection Evaluation

In this chapter the implementation and evaluation for the contextual anomaly detection framework (CADF) will be presented. Specifically, the chapter will describe the implementation of the three major components of the CADF: **content detection, context detection,** and **build clusters**. Then, the implementation will be evaluated against several datasets provided by Powersmiths, a company specializing in environmental sensoring for sustainable buildings. The provided datasets will have injected anomalies in their test data, as well as existing anomalies noted by the Powersmiths employees. This chapter will also provide a discussion on how well the CADF identifies the anomalies, both injected and existing.

This chapter will be organized into three sections. The first section will provide an overview of the implementation, including a discussion on how the components were implemented. The second section will discuss the evaluation of the implementation on the sensor datasets, as well as provide a comparison with other state of the art works. Finally, the last section will summarize the implementation and evaluation for the CADF. The preliminary results for this work were presented in a conference paper. [26]

4.1 Implementation

The primary case study the CADF can be applied is in *Big Sensor Data* applications. These are scenarios whereby a large number of independent sensors stream data to a centralized repository at short time intervals. For example, a building may possess hundreds of electrical sensors, each streaming their current usage information every few seconds. Figure 4.1 illustrates a general sensor streaming application. Utility companies and companies such as Powersmiths, that provide sensoring services to buildings, may then have hundreds, thousands, or even tens of thousands of clients streaming data to their central repository. For such a company, handling this *Big Data* efficiently and effectively in real-time becomes a difficult task. Additionally, there is generally meta-information associated with the sensors, or the buildings they are installed in, which further enhances the computational complexity of handling the data, but also enhances the contextual information associated with the sensors. Due to this, the implementation and evaluation of the CADF will focus on the streaming sensor case study. This section will be broken down into three sub-sections. First, an introduction to the streaming sensor scenario will be provided. Second, the implementation for the components will be outlined. Finally, the implementation for the offline, MapReduce, implementation will be presented.

4.1.1 Big Sensor Data

The coming sections of this chapter will make use of the term *Big Sensor Data*. Here we define *Big Sensor Data* as having the following attributes:

1. The data is streamed constantly at an average rate of 1 write/second to a central data storage location. Depending on the sensor service provider this rate can vary widly from frequencies less than one second, to upwards of 15 minutes.

2. The data storage holds data from a large number of sensors, with each sensor sharing attribute 1.

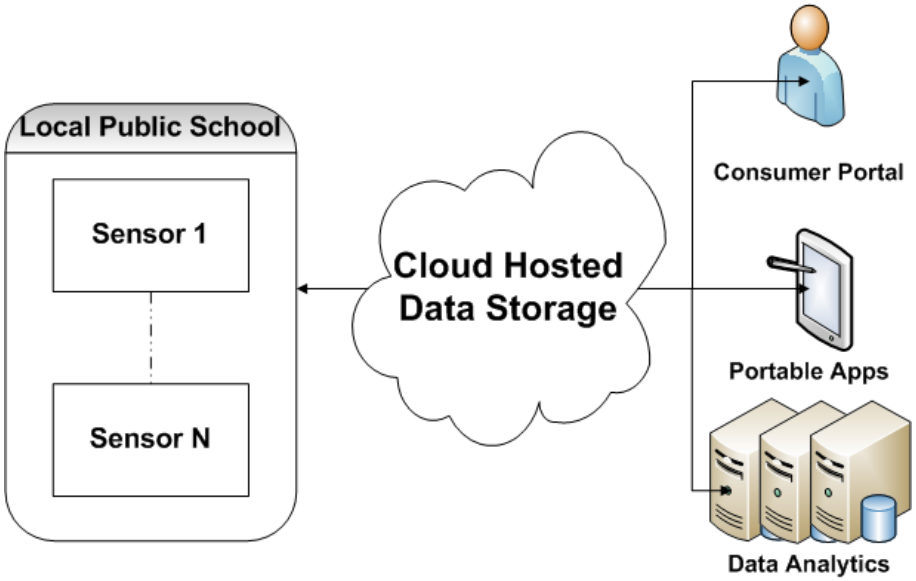


Figure 4.1: Big Sensor Data Case Study

3. The data storage holds data from a wide *variety* of sensors, with each sensor sharing attribute 1. That is, the sensoring provider may have sensoring devices for electricity, hydro, gas, emissions, and waste. These disparate sensoring devices are stored together in the data storage location.

4. Meta-information is associated with the sensors, or buildings the sensors are located in. More importantly, the meta-information is fluid, i.e. the data storage schema can be updated at any time.

The definition provided for *Big Sensor Data* is in-line with standard definitions for *Big Data* itself, as provided in Section 2. The definition provided here describes a more specific view over the standard *Big Data* definition; however, the major attributes associated with *Big Data* are retained: volume, variety, and velocity. Volume is included as there can be up to tens of thousands of sensors streaming data to the central repository at any given time interval. Variety is included as the central repository may include sensor devices for a wide range of uses. Velocity is included as the data is streamed constantly at short intervals. Additionally, the requirement of velocity exists to respond to the sensor data in real-time.

Sensoring has become a prominent feature in many businesses that strive to be more eco-friendly. As a result, a number of companies have emerged to provide environmental sensoring networks for buildings that track information including: electricity, hydro, gas, and other emissions. One such company is Powersmiths, located in Brampton, Ontario, Canada and specializing in providing energy metering and monitoring services through their proprietary application *Windows on the World*. Currently, Powersmiths has thousands of customers across the globe, each harboring tens to hundreds of sensors, all feeding back to a central repository at intervals ranging from seconds to tens of minutes. The goal of these sensoring companies, including Powersmiths, is to provide their clients with a system that can reduce energy waste, thus improving the environment for future generations. To do this, Powersmiths provides analytical features for their clients, normally in a post-processing implementation, to inform them of optimizations or problems within their building. A possible future work is to include real-time analytics, thus enhancing their toolbox they provide to their customers. As a result, Powersmiths has provided access to some historical sensoring data for use in evaluating the CADF. Table 4.1 illustrates an overview of a common schema in their database.

Feature	**Domain**
Time	DD/MM/YYYY HH:MM:ss
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]

Table 4.1: Sensor Dataset and Corresponding Domains

Powersmiths also stores meta-information associated specifically with the locations of sensors. Table 4.2 illustrates a breakdown of some of the meta-information that is stored with different sensors; each row represnts a single sensor. The evaluation section will concretely define the dataset, the features, and the domains, used in evaluation a particular component of

the CADF. Table 4.2 illustrates some important attributes that can be used as contextual information by the CADF. For example, the **build clusters** component may cluster sensors based on their **associated** meta-information, as this is a good indication of the type of sensor it is. However, including the **unit** meta-information with the **associated** meta-information may be important in differentiating between sensors that both read electricity usage, but for different reasons, i.e. lighting versus heating, ventilation and air-conditioning (HVAC). Further, when the **context detector** evaluates the sensor value, the room information itself may become an important characteristic. For example, the *research and development* office may have a much higher average energy usage than the *executive office* room. The relationships described here are simple; however, the **context detector** will learn more complex relationships within the different context, thus deploying a more accurate model for handling these contexts.

Location	Sensor Location	Associated	Room Information	Unit
Main	Total Usage	HVAC	Total	kWh
Main	1st Floor	HVAC	Administrative Office	kWh
Main	1st Floor	Lighting	Administrative Office	kWh
Main	1st Floor	HVAC	Research and Development	kWh
Main	2nd Floor	Lighting	Forge	kWh
Main	2nd Floor	HVAC	Forge	kWh
Main	2nd Floor	GHG Emissions	Forge	kWh

Table 4.2: Sensor Associated Meta-Information

4.1.2 Components

Chapter 3 introduced the three components that are part of the CADF: **content detection, context detection**, and **build clusters**. The first two components were implemented using the object-oriented paradigm in Java as interfaces that can be implemented independently by the application depending on the required underlying algorithm. For this research, classes were created for the univariate and multivariate Gaussian predictors implementing this interface. Future work can implement new classes using the same interface to have different components.

It is worth mentioning that the choice of univariate Gaussian and multivariate Gaussian predictors for the two predictor components is due to the underlying Gaussian distribution seen in the Big sensor Data datasets used in this evaluation. A histogram for the frequency of values seen for one of the datasets is shown in Figure 4.2. The data storage itself was extracted manually to separate .csv files for use by the Java application.

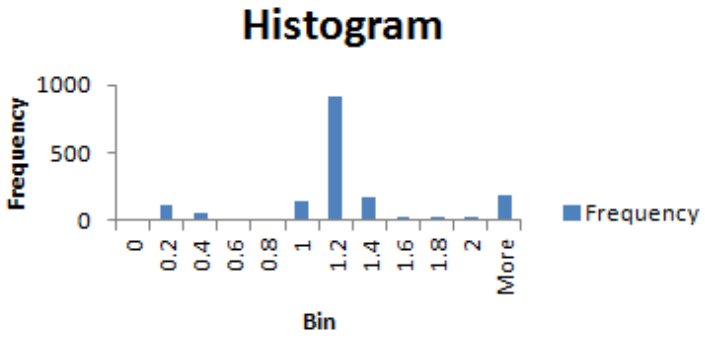


Figure 4.2: Frequency Histogram for Evaluation Data

The implementation for the `Univariate Gaussian Predictor` is based on two primary functions: building the real-time model, and evaluating the real-time model. A code snippet for building the real-time model can be found in Listing 4.1, and a code snippet for evaluating the real-time model can be found in Listing 4.2. These two implementations are based on the equations defined in Section 3.2. These functions are called as part of the main Java application. Therefore, to update the model used by the content detector, one only needs to define two functions: one to build the model, and one to evaluate the model. Due to the modularity of the application, these new functions just replace the existing two functions, without effecting the rest of the implementation. These functions directly correlate to the equations found in Chapter 3: Equation 3.1, Equation 3.2, and Equation 3.3.

.

Listing 4.1: Code Snippet to Build the Real-Time Model

```
double intermediate = 0;
for (int i = 0; i < m; i++) {
       intermediate += singleDataTrain.get(i);
}
mu = (1 / (double) m) * intermediate;
intermediate = 0;
for (int i = 0; i < m; i++) {
       intermediate += Math.pow((singleDataTrain.get(i) - mu), 2);
}
sigma = (1 / (double) m) * intermediate;
```

.

Listing 4.2: Code Snippet to Evaluate the Real-Time Model

```
double predictedValue = 0.0;
predictedValue = (1 / (Math.sqrt(2 * Math.PI * sigma)))* Math.exp(-1 *
    (Math.pow((value - sigma), 2) / (2 * sigma)));

if (predictedValue < epsilon)
       return true;
return false;
```

The `Multivariate Gaussian Predictor` was developed in a similar way to its Univariate counterpart. Listing 4.3 defines the function to build the multivariate predictor. This function is partly based on the results from building the clusters using MapReduce. Therefore, the code shown in Listing 4.3 is called *n* number of times, where n is the number of clusters determined by the k-means clustering algorithm. Again, the implementation is based on the equations defined in Section 3.3. The separation of concerns between all modules of the application is retained. When a different model is required for the context detector, one only needs

to define a function to build the new model, and evaluate the new model. Like the univariate model, the multivariate gaussian implementation directly correlates to the equations defined in Chapter 3: Equation 3.5, Equation 3.6, and Equation 3.7.

.

Listing 4.3: Code Snippet to Build the Multivariate Gaussian Model

```
//Grab the clusters from the kMeans clustering algorithm
int[] assignments = kMeans.getAssignments();
//Get the subset of the data associated with that cluster
int z = 0;
for (int clusterNum : assignments) {
       dataIn.get(clusterNum).add(data.get(z));
       z++;
}
//Determine the parameters for the model for this cluster
for (int a = 0; a < k; a++) {
       double intermediate = 0;
       for (int j = 1; j < numberOfFeatures; j++) {
              intermediate = 0;
              for (int i = 0; i < dataIn.get(a).size(); i++) {
                     intermediate +=
                         Double.parseDouble(dataIn.get(a).get(i)[j]);
              }

              mu_array[a][j] = (1 / (double) dataIn.get(a).size()) *
                  intermediate;
       }

       for (int j = 1; j < numberOfFeatures; j++) {
```

```
                intermediate = 0;
                for (int i = 0; i < dataIn.get(a).size(); i++) {
                        intermediate +=
                            Math.pow((Double.parseDouble(dataIn.get(a).get(i)[j])
                            - mu_array[a][j]),2);
                }
                sigma_array[a][j] = (1 / (double) dataIn.get(a).size()) *
                    intermediate;
        }
}
```

.

Listing 4.4: Code Snippet to Evaluate the Multivariate Gaussian Model

```
double predictedValue = 1.0;
for (int j = 1; j < numberOfFeatures; j++) {
        predictedValue *= (1 / (Math.sqrt(2 * Math.PI*
            sigma_array[cluster][j])))* Math.exp(-1*
            (Math.pow((Double.parseDouble(value[j]) -
            sigma_array[cluster][j]),2) / (2 * sigma_array[cluster][j])));
}
 if (predictedValue < epsilon) {
        return true;
}
return false;
```

4.1.3 Virtualized Sensor Stream

The implementation for the CADF 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. From there, each sensors dataset was independent, with its own defined set of write frequency rules. When a new sensor wanted to write to the content detector, the value would be written into a `Queue` data structure. The queue would then be incrementally evaluated and removed. A code listing for this process can be found in Listing 4.5.

.

Listing 4.5: Code Snippet for Virtualized Sensor Stream

```
//Instantiate Queue
Queue virtualQueue = new LinkedList();
...
//Concurrently write new sensor values to the queue
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);
       }
}
```

4.2 Evaluation

The CADF evaluation is based on the implementation described in Section 4.1. The evaluation will be described in the context of several scenarios. Specifically, this section will describe the content and context detection for two streaming sensor datasets: the first corresponds to an HVAC system in a building, and the second corresponds to the lightning electrical usage in a building. Additionally, a comparison of the results of the initial findings without random positive detection, to running the algorithm with random positive detection will be shown. Finally, a comparison between the proposed CADF with an existing, offline, anomaly detection algorithm will be shown.

4.2.1 Dataset 1: Content Detection

The first *Big Sensor Data* dataset to be evaluated is one storing HVAC sensors readings. The dataset consists of four sensors; their meta-information is shown in Table 4.3. The dataset consists of four sensors that are recording HVAC readings, in kilowatt-hours (kWh). The sensors are distributed across the building on two floors, and four locations. An example of the readings at a few time intervals is shown in Table 4.4. Table 4.4 also illustrates some context that can be extracted from the *Time* attribute. Specifically, two additional features are included in the meta-information for each sensor reading: *Day* and *TimeOfDay*. The first dataset consists of 101,384 tuples. The dataset was split into two components: 85% of the dataset was used in training the detection models, and 15% of the dataset was used in building the simulation, test, environment.

Location	Sensor Location	Associated	Room Information	Unit
Main	1st Floor	HVAC	Administrative Office	kWh
Main	1st Floor	HVAC	Research and Development	kWh
Main	2nd Floor	HVAC	Executive Office	kWh
Main	2nd Floor	HVAC	Forge	kWh

Table 4.3: Sensor Dataset 1: HVAC

DateTime	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Day	TimeofDay
04/06/2010 12:00	37.9	4.9	1.1	3.9	1	2
04/06/2010 13:00	36.8	2.8	1.9	2.4	1	2
04/06/2010 14:00	36.7	3	2	3	1	2

Table 4.4: Dataset Example

The first component to be evaluated is the content detection. A successful content detection implementation should allow anomalies to be detected in *real-time*, and still successfully detect anomalies. In testing the application's content detection using the test data, the content detector was able to find 23 anomalies. These 23 anomalies are considered to be point anomalous, in other words, they are anomalous based on their sensor reading alone. This is consistent with an average number of anomalies for such a streaming system, as it identified 0.0015% of the values as anomalous. The results of the content detection are shown in Figure 4.3.

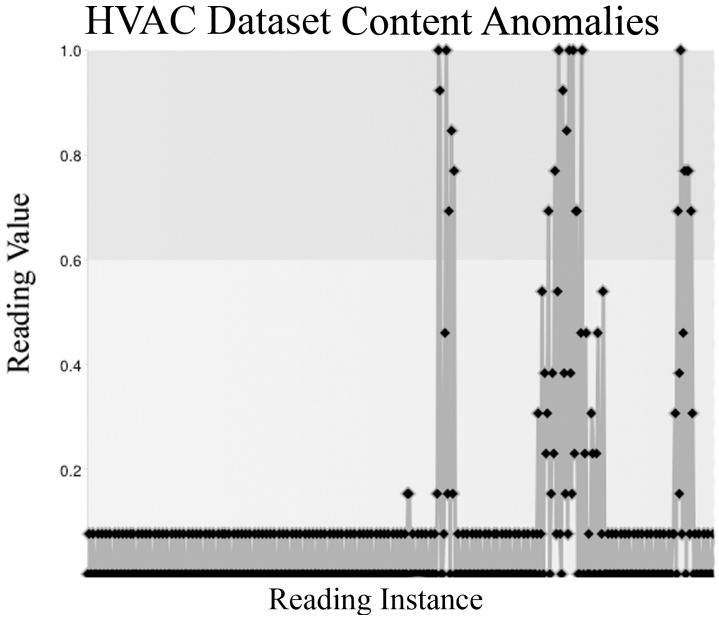


Figure 4.3: Content Detection for Dataset 1: the black diamonds represent a sensor reading instance, and the grey lines connect continuous reading instances. The highlighted portion at the top represents those reading instances identified as anomalous with respect to content.

4.2.2 Dataset 1: Context Detection

The second component to be evaluated for the CADF is the context detection. The first component to test is building the clusters, *sensor profiles* for the dataset. For dataset 1, there were two clusters that emerged for the set of five sensors. The first cluster included sensor 1 and 5. The second cluster included sensors 2, 3 and 4. After determining the clusters, two multivariate gaussian predictors were trained, one for each cluster. The context detector determined that two of the content anomalies were false positives. One immediate conclusion is that the context detection provided additional insight to the values determined to be anomalous by including information such as *Day of the Week* and *Time of Day* into the prediction calculation.

4.2.3 Dataset 1: Discussion

The results of the CADF on dataset 1 are promising. The framework efficiently determined content based anomalies in real-time using the `Univariate Gaussian Predictor`, while reducing the number of false positives generated using the `Multivariate Gaussian Predictor`. Details on the running time for these algorithms are shown in Table 4.5. To give context to these numbers: the dataset was evaluated on a quad-core laptop (2.1 GHz), running Windows 7 with 8 GB of ram and 8 virtual threads. As shown in the table, the CADF can successfully manage to detect anomalies in real-time, using an average of 933ns per content evaluation. The longest running components were: initially reading in the bulk data from the dataset (in this case, a csv file), and training the two models offline.

Component	Time
readCSV	3.491 sec
buildContextModel	0.136 sec
buildRealtimeModel	0.021 sec
checkRealTime (average)	933 ns

Table 4.5: Dataset 1 Running Time Results

The CADF was able to identify all the anomalous values that were injected by the author

into the dataset. Table 4.6 portrays the anomalous entries entered by the author. In addition to successfully detect the anomalous values injected by the author, the algorithm also identified anomalies that were already known to Powersmiths. Table 4.6 also shows the contextual anomalies that were cleared, below the double horizontal line. In particular, the context predictor found the two values listed in the table as non-anomalous when considered with the TimeOfDay and Day contextual attributes. Looking deeper, it can be extrapolated that these values were considered non-anomalous as though the fourth sensor had extremely low readings (0 and 0.1 kWh respectively), this occurred on a weekend during the morning hours. This particular sensor is for the Forge location at Powersmiths which is normally offline at this period during the week.

Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 4**	**Sensor 5**	**Day**	**TimeofDay**
1.3	0.5	0	0	3	3	1
21.9	22.5	21.5	22	6	2	1
0	0	-1.4	2.3	7	1	3
1.3	0.7	0.8	0	6	3	2
1.1	0.4	0.6	0.1	6	3	0

Table 4.6: Anomalous Examples for Dataset 1

4.2.4 Dataset 2: Content Detection

The second *Big Sensor Data* dataset to be evaluated is the **Temperature** sensors, located in an office building; a subset of the data is shown in Appendix A. The dataset consists of five sensors; their meta-information is shown in Table 4.7. 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 4.8. The second dataset is marginally smaller, consisting of 26,943 readings. For testing and evaluation purposes, the dataset was

split into a training dataset consisting of 85% of the data, and a test dataset used the build the simulation with the remaining 15%.

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

Table 4.7: Sensor Dataset 2: Temperature

DateTime	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Day	TimeofDay
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

Table 4.8: Dataset Example

Synonymous to dataset 1, the first component to evaluate is the content detection. In testing the CAFD's content detection using the test data, the content detector was able to find 243 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 a failed sensor. These results are shown in Figure 4.4. More discussion on this will occur after addressing the context detection component.

4.2.5 Dataset 2: Context Detection

The second major component to be evaluated for the CADF is the context detection. To perform context detection, the CADF first needs to determine the sensor profiles for the temperature sensors. For dataset 2, the CADF 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

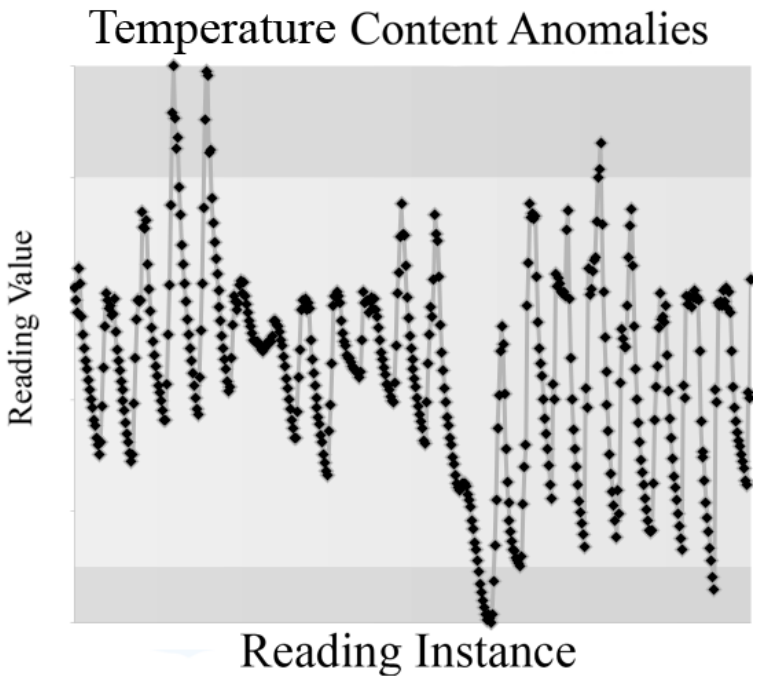


Figure 4.4: Content Detection for Dataset 2: the black diamonds represent reading instances. The top and bottom highlighted portions in grey identify reading instances that were considered anomalous with respect to their content.

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* 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 4.5. The figure shows the distribution of context anomalies found within the content anomalies.

4.2.6 Dataset 2: Discussion

The results for dataset 2 were also promising. The CADF 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 4.9. From the table: the CADF can successfully manage to detect anomalies in real-time, only using an average of 933ns per content evaluation.

Component	Time
readCSV	1.145 sec
buildContextModel	2.343 sec
buildRealtimeModel	0.021 sec
checkRealTime (average)	933 ns

Table 4.9: Dataset 2 Running Time Results

When given this dataset, Powersmiths mentioned that there were lengths of time when various temperature sensors failed at their headquarters. The CADF was able to successfully identify these anomalous readings, as shown below the double-line break in Table 4.10. 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 4.10.

Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Day	TimeofDay
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

Table 4.10: Anomalous Examples for Dataset 2

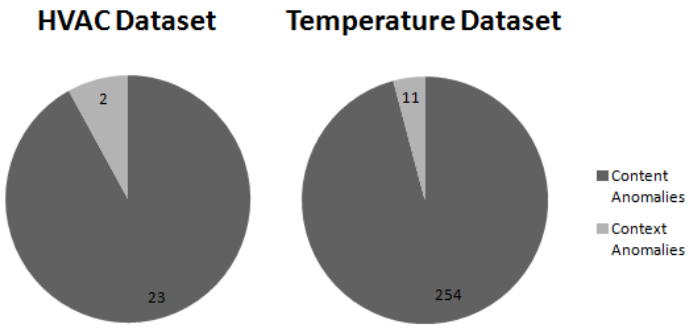


Figure 4.5: Context Detection for Datasets 1 and 2

4.2.7 Random Positive Detection

The previous CADF components did not include the random positive detection component, as presented in Chapter 3. The random positive detection component was introduced to address the issue of the content detection predictor not identifying contextual anomalies that are *not* also content anomalies. That is, the random positive detection is included to reduce the number of *false negatives*. Listing 4.6 shows what needs to be modified in the codebase to include randomized positive detection. Note that this is a basic approach to introducing randomized positive detection: it would be more appropriate, and a potential future work, to skew the randomness for values that are close to the threshold of being considered point anomalous. Unfortunately, the results for the preliminiary random positive detection component did not change the outcome. For dataset 2, there were still 11 contextual anomalies detected.

.

Listing 4.6: Code Snippet for Including Randomized Positive Detection

```
while(true) {
        if(virtualQueue.poll() != null) {
                String val[] = virtualQueue.remove();
                boolean isAnomalous = checkValueWithRealTime(Double
```

```
                                    .parseDouble(val[1]));
            if (isAnomalous || randomGen()==1) {
                    anomalyArray.add(value);
            }
        }
}
```

4.2.8 Comparison With Other Algorithms

The previous evaluation sections attempted to provide insight into the validity of the proposed CADF with respect to real-world data. This section will further compare the results of the CADF to results from the popular **R** project for statistical computing. In particular, the CADF will be compared with the *outliers* toolbox provided by the **R** project. Figure 4.6 shows the first step in computing anomalies with R. The figure outlines the density distribution of the values for dataset 2, with which the CADF results will be compared. Figure 4.7 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 4.7, there are anomalous values in both sections. A summary of the comparison between the CADF and the **R** outliers package is shown in Table 4.11.

4.3 Summary

In this chapter the implementation and evaluation for the CADF was presented. The CADF was evaluated against several real-world sensor datasets, including datasets for HVAC electricity systems and building temperature systems. The chapter also presented important implementation details and considerations used when deploying the CADF; in particular, a discussion on the virtualized sensor stream to emulate real-world streaming sensor applications was shown.

Concept	Discussion
Running time	The biggest difference between the CADF and the R outliers package is in the running time of both algorithms. One of the major design considerations for the CADF 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 the CADF runs and parses the data in less than seconds.
Anomalies found	Both the CADF and R outliers package detected a number of equivalent anomalies. In particular, the huge list of anomalies pertaining to the failed sensors identified apriori by Powersmiths were found by both detectors. Further, the CADF detected all of the anomalies detected by the R outliers package, including those that were contextually cleared.

Table 4.11: Results Comparison between the CADF and R Outliers Package

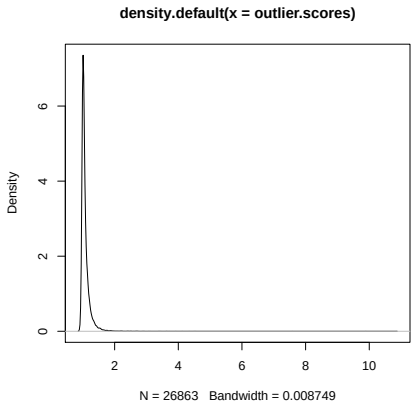


Figure 4.6: Average Density of Data Found by R

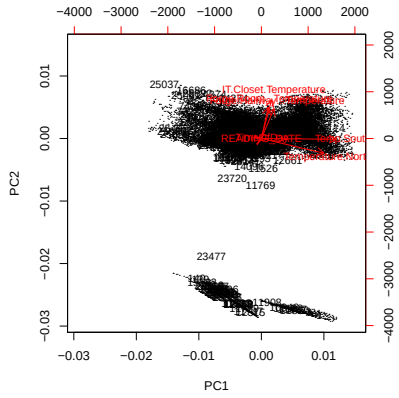


Figure 4.7: R Results: Graphical Representation

Finally, a set of discussions on the validity of the proposed CADF occurred for each of the datasets, and when comparing the results fo the CADF against a statistical package provided by the R suite of statistical tools.

Chapter 5

Conclusion and Future Work

This chapter provides a summary of the conclusions made based on the implementation and evaluation of the Contextual Anomaly Detection Framework (CADF) presented in this thesis. Additionally, a description of possible future works for the CADF will be outlined.

5.1 Conclusion

The work in this thesis 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 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 thesis also contributes a MapReduce approach to determine the clusters needed in the contextual anomaly 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. In this case, the Reduce() operation only needs to consider the centroids of the initial Map() operations, rather than the thousands of tuples used to build those centroids.

This thesis also provided implementation and evaluation for a proof of concept of the CADF. Specifically, the thesis introduced implementation-level details regarding:

- Development and deployment of the virtualized streaming sensor network. This also included considerations for handling multiple sensors streaming new values into the content detector simultaneously.

- Development of the three major components of the CADF: the content detector, the context detector, and the clustering component. A discussion on the loose coupling between components was also presented, with further dialogue on the merits of the modularity of each component.

- A brief overview of the MapReduce component was also presented. This was used in speeding up the training of the clustering algorithm for use in the context detector.

The evaluation of the CADF was also discussed based on the implementation details provided in this thesis. The evaluation of the CADF was performed using two sets of Big sensor Data; one for a set of typical HVAC electricity sensors in a building, and one for a set of temperature sensors throughout rooms within a building. The evaluation provided some conclusions of the work:

- The CADF was able to positively detect, in real-time, content based anomalies for both datasets. Further, the context detector was able to determine some anomalies that should not be considered anomalous when evaluated with respect to context in addition to content.

- The CADF was able to positively detect anomalies that were determined apriori by Powersmiths, the provider of the datasets. In particular, the CADF determined a large set of anomalous readings which occurred when Powersmiths indicated a massive sensor failure in their system.

- The CADF performed competitively with the R outlier statistical package. In fact, the CADF performed in real-time, in comparison with the batch approach provided by the R outlier package. Additionally, the CADF determined the same values to be anomalous as the R statistical package.

Anomaly detection is an important aspect of many real-time, and offline systems. Businesses rely on their systems to perform efficiently and effectively without failures. Anomaly detection algorithms and frameworks exist to ensure that these systems continue to perform by identifying anomalies so the business or system can react effectively. As data continues to grow, there is a need for anomaly detection algorithms, and machine learning algorithms in general, to be able to scale and still provide usefulness to their system. The CADF proposed in this work provides one such framework to cope with the difficulties of Big Data in the domain of streaming sensors, using a hierarchical, modular, approach that ensure the computationally expensive, accurate, models are only evaluated on a small subset of the overall data. This paradigm shift in the way algorithms are developed for Big Data serves as a starting point to address Big Data challenges for computationally expensive algorithms.

5.2 Future Work

As previously described, one of the major goals of the CADF was to remain modular and scalable for future works. This was not only to ensure the framework could scale to Big sensor Data applications, but also to ensure that the work remain viable as newer research is completed. As a result, there are several areas of future work that would be interesting to explore:

- The dataset considered in the evaluation section is only one type of Big Data; that is, a *tall* dataset. It is important to consider the other common type of application, *wide* datasets. These are attributed by a large number of features horizontally, with a smaller number of records. Extending the CADF to this area of future work is interesting as the CADF should perform even better for a horizontal Big Data application. 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. However, the contextual detector thrives on the number of features associated with each record; these features give more *context* to the data. While it is true that adding features which contribute little to the context is a concern, there is still a good probability that these new features will contribute positively to the CADF. Additionally, adding new features will not effect the content detector; therefore, the real-time attribute of the CADF will be preserved.

- More dicussion and exploration related to the random positive detection component of the CADF should be explored. While there were some promising results, the random positive detection did not contribute as prevalently as expected. For most cases, the addition of the random positive detection did little to reducing the number of false positives. A future work here is to provide a more intelligent random positive detection mechanism. For example, in addition to truly randomly sending values to the contextual detection algorithm, one can also send values which were on the cusp of being anomalous based on the content.

- The CADF has been tested in the simulated, virtualized, sensor streaming environment. While the implementation closely mimics the real-world scenario of many sensors attemping to write to a central repository concurrently, it is certainly not the same as evaluating the work in a fully streaming environment. Thus, a future work would be to implement the CADF within a working business environment that is streaming live

data to the central repository. The addition of such an evaluation would provide further strength into the framework's ability to detect anomalies in real-time.

- In addition to implementing the framework in a real business environment, the framework could also be integrated with decision making systems within such an 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. For example, should it be determined that an electrical outlet is being consistently active over a holiday period, the event processing framework can remotely, and dynamically, shut off the electrical outlet until the holiday period ends. On a smaller scale, the decision making framework could simply send out notifications to decision makers when the CADF provides it with anomaly information.

- The modularity of the CADF allows further components to be added, or updated, in the framework. This introduces two such avenues of future work: first, the modules that are existing can be modifed and updated with other types of algorithms. The CADF currently uses univariate and multivariate Gaussian predictors for the content and context detectors. However, perhaps research will be presented that shows a Bayesian predictor performs better in parallel implementations. The univariate module can then be replaced by the one-step predictor, while keeping the rest of the framework intact. A second extension would be to implement additional modules to the framework itself. For example, a semantic detection algorithm module can be included as a third tier in the frameworks hierarchy. Perhaps this will reduce the number of false positives even further than the contextual detector. The drawback of the semantic detection module is that it would need to be used in an application where a semantic ontology can be defined. Some applications inherently have these, specifically many Web-based applications, and so adding this module may further enhance the performance of the CADF.

While there is room for many possible future works for the CADF, this thesis serves as a starting point and proof of concept for one method to bring anomaly detection from traditional datasets to *Big Data*. The future extensions for this work are possible because of the underlying modular, scalable architecture presented to handle *Big Data*, especially in domains such as streaming sensors where a high velocity of new values are written frequently. The future works also stresses the importance of changing the method in which we deploy algorithms from traditional datasets to *Big Data*.

Bibliography

[1] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the eight ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.

[2] V.S. Agneeswaran, P. Tonpay, and J. Tiwary. Paradigms for realizing machine learning algorithms. *Big Data*, 2013.

[3] A. AlEroud and G. Karabatis. A contextual anomaly detection approach to discover zero-day attacks. In *Cyber Security (CyberSecurity), 2012 International Conference on*, pages 40–45, Dec 2012.

[4] Apache. Apache hadoop, February 15th, 2014.

[5] Apache. Apache mahout.

[6] Apache. Apache spark.

[7] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[8] S. Ayhan, J. Pesce, P. Comitz, D. Sweet, S. Bliesner, and G. Gerberick. Predictive analytics with aviation big data. In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2013*, pages 1–13, April 2013.

[9] M. Barnett, B. Chandramouli, R. DeLine, S. Drucker, F. Fisher, Goldstein. J., P. Morrison, and J. Platt. Stat!-an interactive analytics environment for big data.

[10] P. Bhatotia, A. Wieder, R. Rodrigues, U.A. Acar, and R. Pasquin. Incoop: MapReduce for incremental computations. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 7. ACM, 2011.

[11] D. Borthakur. *The Hadoop Distributed File System: Architecture and Design*. 2014.

[12] Y. Bu, B. Howe, M. Balazinska, and M.D. Ernst. Haloop: Efficient iterative data processing on large clusters. *Proc.VLDB Endow.*, 3(1-2):285–296, sep 2010.

[13] E. Burns. Limited role for big data seen in developing predictive models, October 2013.

[14] H. Chan, P. Chou, S. Duri, H. Lei, and J. Reason. The design and implementation of a smart building control system. In *e-Business Engineering, 2009. ICEBE '09. IEEE International Conference on*, pages 255–262, Oct 2009.

[15] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[16] B. Chandramouli, J. Goldstein, and A. Quamar. Scalable progressive analytics on big data in the cloud. *Proceedings of the VLDB Endowment*, 6(14):1726–1737, 2013.

[17] T. Condie, N. Conway, P. Alvaro, J.M. Hellerstein, K. Elmeleegy, and R. Sears. MapReduce online. In *Proceedings of the 7th USENIX conference on Networked Systems Design and Implementation*, page 21, Berkeley, CA, USA, 2010. USENIX Association.

[18] B. Dalessandro. Bring the noise: Embracing randomness is the key to scaling up machine learning algorithms. *Big Data*, 1(2):110–112, 2013.

[19] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[20] J. Fan, F. Han, and H. Liu. Challenges of big data analysis. *National Science Review*, in press, February 06 2014.

[21] S. Ghemawat, H. Gobioff, and S-T. Leung. The google file system. In *ACM SIGOPS Operating Systems Review*, volume 37, page 29, New York, New York, USA, dec 2003. ACM Press.

[22] K. Grolinger, M. Hayes, W.A. Higashino, A. L'Heureux, D. Allison, and M.A.M. Capretz. Challenges for MapReduce in Big Data. In *Proceedings of the upcoming 10th IEEE World Congress on Services (SERVICES)*, July 2014.

[23] K. Grolinger, W.A. Higashino, A. Tiwari, and M.A.M. Capretz. Data management in cloud environments: NoSQL and NewSQL data stores. *Journal on Cloud Computing: Advances, Systems, and Application*, 2, 2013.

[24] A. Hall, O. Bachmann, R. Bassow, S. Garceanu, and M. Nunkesser. Processing a trillion cells per mouse click. *Proceedings of the VLDB Endowment*, 5(11):1436–1446, 2012.

[25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer, 2009.

[26] M. Hayes and M.A.M. Capretz. Contextual anomaly detection in big sensor data. In *Proceedings of the upcoming 3rd IEEE International Congress on Big Data*. IEEE, July 2014.

[27] Z. He, J.Z. Huang, X. Xu, and S. Deng. Mining class outliers: Concepts, algorithms and applications. In Qing Li, Guoren Wang, and Ling Feng, editors, *Advances in Web-Age Information Management*, volume 3129 of *Lecture Notes in Computer Science*, pages 589–599. Springer Berlin Heidelberg, 2004.

[28] J. Heer and S. Kandel. Interactive analysis of big data. *XRDS: Crossroads, The ACM Magazine for Students*, 19(1):50–54, 2012.

[29] D.J. Hill and B.S. Minsker. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environ. Model. Softw.*, 25(9):1014–1022, September 2010.

[30] D.J. Hill, B.S. Minsker, and E. Amir. Real-time bayesian anomaly detection in streaming environmental data. *Water Resources Research*, 45(4), 2009.

[31] T. Hill and P. Lewicki. *STATISTICS: Methods and Applications*. StatSoft, Tulsa, OK, 2007.

[32] E. Junqué de Fortuny, D. Martens, and F. Provost. Predictive modeling with big data: Is bigger really better? *Big Data*, 2013.

[33] J. Kittler, W. Christmas, T.D. Campos, D. Windridge, F. Yan, J. Illingworth, and M. Osman. Domain anomaly detection in machine perception: A system architecture and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints):1, 2013.

[34] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[35] S.B. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Data preprocessing for supervised learning. *International Journal of Computer Science*, 1(2):111, 2006.

[36] Y. Kou and C-T. Lu. Spatial weighted outlier detection. In *Proceedings of SIAM Conference on Data Mining*, 2006.

[37] 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*, pages 282–287, 2013.

[38] J. Lin. MapReduce is good enough? if all you have is a hammer, throw away everything that's not a nail! *Big Data*, 1(1):28–37, mar 2013.

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

[40] G. Malewicz, M.H. Austern, A.J.C. Bik, J.C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

[41] S. Melnik, A. Gubarev, J.J. Long, G. Romer, S. Shivakumar, M. Toltonm, and T. Vassilakis. Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, 3(1-2):330–339, 2010.

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

[43] F. Ohlhorst. *Big data analytics: turning big data into big money*. Wiley, Hoboken, N.J, USA, 2013.

[44] Committee on the Analysis of Massive Data; Committee on Applied, Theoretical Statistics; Board on Mathematical Sciences, Their Applications; Division on Engineering, and Physical Sciences; National Research Council. *Frontiers in Massive Data Analysis*. The National Academies Press, 2013.

[45] S. Rajasegarar, C. Leckie, and M. Palaniswami. Anomaly detection in wireless sensor networks. *Wireless Communications, IEEE*, 15(4):34–40, 2008.

[46] S. Sagiroglu and D. Sinanc. Big data: A review. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, pages 42–47, May 2013.

[47] 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*, pages 491–496, 2013.

[48] X. Song, Mingxi Wu, C. Jermaine, and S. Ranka. Conditional anomaly detection. *Knowledge and Data Engineering, IEEE Transactions on*, 19(5):631–645, May 2007.

[49] S.N. Srirama, P. Jakovits, and E. Vainikko. Adapting scientific computing problems to clouds using mapReduce. *Future Generation Computer Systems*, 28(1):184–192, 2012.

[50] 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*, pages 13–24, 2010.

[51] 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*, pages 751–759, 2012.

[52] M. Zinkevich, M. Weimer, A.J. Smola, and L. Li. Parallelized stochastic gradient descent. 4(1):4, 2010.

Appendix A

Big Sensor Data Examples

Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
3	18.49167	20.025	24.79167	6.96667	8.70833	1
3	18.50833	20.01667	24.78333	7.06667	8.96667	1
3	18.55833	20.03333	24.8	7.10833	9.04167	1
3	18.64167	20.05	24.79167	7	8.95	1
3	18.69167	20.03333	24.8	6.01667	6.1	2
3	18.96667	20.01667	24.76667	5.04167	3.50833	2
3	18.76667	19.96667	24.76667	4.55833	2.875	2
3	18.64167	19.80833	24.61667	4.65833	2.40833	2
3	18.525	19.63333	24.53333	4.98333	1.75	2
3	18.08333	19.40833	24.35	4.54167	0.68333	2
3	18.50833	20.26667	24.475	3.85	-0.49167	2
6	18.38333	20.14167	24.45833	3.6	-1.54167	0
6	18.225	20.09167	24.45833	3.13333	-2.66667	0
6	18.21667	19.84167	24.49167	2.8	-3.41667	0
6	18.38333	20.34167	24.475	2.475	-4.1	0
6	18.38333	20.30833	24.43333	2.21667	-5.03333	0
6	18.45833	20.225	24.35833	1.66667	-5.33333	0
6	18.29167	19.325	24.23333	0.64167	-5.69167	0
6	18.575	20.25833	24.36667	0.70833	-5.68333	0
6	18.125	19.70833	24.30833	-0.00833	-5.24167	0
6	18.81667	20.55	24.43333	1.53333	-4.7	1
6	0	0	0	3.05	-4.36667	1
6	0	0	0	4.88333	-3.65833	1
6	0	0	0	5.55	-3.3	0
6	0	0	0	5.91667	-2.18333	1
6	0	0	0	6.25	-1.925	1
6	0	0	0	5.44167	-1.84167	1
6	0	0	0	3.84167	-1.875	1
6	0	0	0	2.54167	-2.04167	2
6	0	0	0	0.10833	-3.46667	2

Curriculum Vitae

Name:	Michael Hayes
Year of Birth	1989
Place of Birth	Toronto, Ontario, Canada
Post-Secondary Education and Degrees:	Western University London, ON 2012-2014 MESc
	Western University London, ON 2008-2012 BESc
Honours and Awards:	NSERC CSG M 2013-2014
	Ontario Graduate Scholarship 2012-2013
	NSERC USRA 2011-2011
	Western Scholarship of Excellence 2008-2009
Related Work Experience:	Teaching Assistant Western University 2012 - 2014
	NoSQL Database Requirements and Design Engineering Powersmiths 2013-2014
	Lawson Security Consultant

Lawson Health Research Institute
2011-2012

Software Engineering Summer Academy Instructor
Western University
2011-2012

Publications:

M. A. Hayes, M. A. M. Capretz, "Contextual Anomaly Detection in Big Sensor Data" in Proceedings of the upcoming 3rd IEEE International Congress on Big Data, Anchorage, Alaska, June 27-July 2, 2014.

K. Grolinger, **M. Hayes**, W.A. Higashino, A. LHeureux, D. Allison, and M.A.M. Capretz, "Challenges for MapReduce in Big Data" in Proceedings of the upcoming 10th IEEE World Congress on Services (SERVICES), Anchorage, Alaska, July 27-July 2, 2014.

S. Wang, W.A. Higashino, **M.A. Hayes**, M. A. M. Capretz, "Service Evolution Patterns" in Proceedings of the upcoming IEEE International Conference on Web Services (ICWS), Anchorage, Alaska, June 27-July 2, 2014.

K. Brown, **M. A. Hayes**, D. S. Allison, M. A. M. Capretz, M. Sazio, R. Mann, (2013) "Fine-grained filtering to provide access control for data providing services within collaborative environments", Concurrency and Computation: Practice and Experience. DOI: 10.1002/cpe.3167.

M. A. Hayes, M. A. M. Capretz, J. Reed, C. Forchuk, (2012) "An Iterative Association Rule Mining Framework to K-Anonymize a Dataset", ASE Science Journal, 1 (4), 179-194.

M. A. Hayes and C.J. Gibson. "Sports Injury Decision Support System - Mobile Application". Poster Presented at: Advances in Health Informatics Conference (AHIC), April 25th, 2012, Toronto, Canada.

K. Brown, **M. A. Hayes**, D. S. Allison, M. A. M. Capretz, R. Mann, "Fine-Grained Filtering of Data Providing Web Services with XACML" in Proceedings of the IEEE WETICE 2012 - Web2Touch Track, Toulouse, France, June 25-27, 2012.